UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Ingeniería Técnica en Informática de Gestión

Esteganografía en código C

Proyecto Final de Carrera

Autor: Iván Redondo Chisvert

Director: Jorge Blasco Alis
Tutor: Sergio Pastrana Portillo

AGRADECIMIENTOS

Si tuviera que agradecer e incluir en este apartado a toda la gente que se lo merece, esta memoria seguramente duplicaría su tamaño.

En primer lugar a Noelia, la persona que me incito a iniciar esta etapa de mi vida y que me acompaño y ayudo desde el primer momento hasta más allá del último.

A mis padres que siempre han estado ahí para apoyarme y ayudarme.

A mis tutores Jorge y Sergio, en especial a Jorge por su comprensión y ayuda desinteresada durante tanto tiempo.

A Zanni y al resto de mis compañeros de clase por su ayuda y por estar ahí tanto en los buenos como en los malos momentos.

Y por último, no puedo dejar de pensar lo orgulloso que se sentiría mi abuelo si me viese en este momento, aquel que no llegó a ver terminar esta etapa de mi vida y que siempre preguntaba: “Los estudios como van, ¿ya lo has terminado?”. Por esto y muchas otras cosas, no puedo olvidarme de él en este momento.

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES:

ÍNDICE DE TABLAS:

1. INTRODUCCIÓN

1.1. LA ESTEGANOGRAFÍA

La **esteganografía,** derivada de la composición de las palabras griegas *steganos* (στεγανος), que significa cubierto u oculto, y *graphos* (γραφος), que significa escritura, es la disciplina en la que se estudian y aplican técnicas que permiten el ocultamiento de mensajes u objetos, dentro de otros, llamados portadores, de modo que no se perciba su existencia. Es una mezcla de artes y técnicas que se combinan para conformar la práctica de ocultar y enviar información sensible en un portador de modo que no sea advertido el hecho mismo de su existencia y envío. De esta forma, un probable intruso ni siquiera sabrá que se está transmitiendo información sensible, la cual sólo podrá ser recuperada por un usuario legítimo que conozca el algoritmo de extracción de la misma.[1]

Aunque la criptografía y la esteganografía suelen confundirse debido a que ambas son utilizadas para proteger información sensible, la forma en la que lo hacen es muy distinta.

La criptografía no oculta la información a un intruso, sino que la cifra o codifica para que le sea ininteligible a este y la esteganografía al contrario, principalmente lo que trata es de ocultar dicha información en un portador para que al intruso le pase inadvertida su existencia.

Sin embargo, es muy común que estas dos disciplinas se utilicen a la vez dando más seguridad al envió de la información sensible. Si el mensaje se cifra antes del esteganografiado, el intruso no solo tendría que detectar la existencia de dicho mensaje, sino que aun detectándolo, lo que obtendría le seria ininteligible.

La esteganografía principalmente se basa en utilizar los límites para distinguir información que tienen los sentidos humanos de la vista y el oído para ocultar ahí la información.

Pero este método no es perfecto, y deja una serie de huellas o pequeños cambios en el medio portador utilizado que mediante las técnicas de estegoanálisis se puede llegar a detectar la existencia de dicho mensaje oculto.

[1] Definición extraída **Esteganografía y Estegoanálisis.** Autores: Jordi Serra y Daniel Lerch.

1.2. TERMINOLOGÍA

Antes de continuar con la explicación del concepto de esteganografía y de sus diferentes técnicas deben de quedarnos clara la terminología general sobre la esteganografía que posteriormente vamos a utilizar:

- **Esteganografía:** (del gr. στεγανος 'oculto' + γραφή 'escritura') Aunque no está en DRAE[2], es formalmente correcta y adecuada a su significado, y alude a la ocultación de un mensaje en otro mensaje (sea texto o imagen)[3].

- **Esteganografiado y desesteganografiado:** Estas definiciones al igual que la anterior, no son acepciones oficiales de la DRAE, pero hacemos un uso libre de las mismas para facilitar la lectura y comprensión del proyecto.

  Esteganografiado: Ocultar la información dentro del portador.

  Desesteganografiado: Obtener la información oculta dentro del portador.

- **Información oculta:** Es la información que se envía de forma secreta u oculta.

- **Portador u Objeto contenedor**: Es la entidad (pista de audio, imagen, vídeo, texto, o en nuestro caso el fichero de código C) que se emplea para portar el mensaje oculto.

- **Estego-objeto**: Se define así al objeto usado como portador una vez que ya se ha esteganografiado la información en él.

- **Estego-clave**: Es la clave utilizada en el proceso de esteganografiado.

- **Adversarios:** Son todos aquellos entes a los que se trata de ocultar la información encubierta. Este adversario puede ser pasivo o activo.

- **Estegoanálisis**: Es ciencia dedicada al estudio de la detección y/o anulación de información oculta en distintas tapaderas, así como la posibilidad de localizar la información útil dentro de la misma (existencia y tamaño).

  Para ellos se pueden llevar a cabo dos tipos de ataques:

  - Ataques pasivos o de detección: El adversario sospecha que se puede estar produciendo una comunicación encubierta y trata de descubrir el algoritmo que se extrae del estego-objeto, pero no trata de modificar dicho objeto.

  - Ataques activos o de anulación: El adversario, además de tratar de hallar el algoritmo de comunicación encubierta, modifica el estego-objeto con el fin de corromper cualquier intento de mensajería subliminal.

[2] Diccionario de la Real Academia Española: http://www.rae.es/recursos/diccionarios/drae
[3] Definición sacada de la consulta vía twiter de Lucia Escapa @Lucia_E el 20 de dic. de 2012 a la RAE: https://twitter.com/raeinforma/status/281759777770594304

- **Invisibilidad perceptiva:** Se refiere al grado en que la información oculta incluida debe pasar inadvertida a los sentidos de todo el mundo menos al destinatario de la misma.

- **Invisibilidad estadística o algorítmica**: Se refiere al grado en que la información oculta es invisible ante estegoanálisis.

- **Robustez:** Es la resistencia que tiene la información oculta ante la manipulación "inocente" del portador de la información (estego-objeto).

- **Seguridad de un método de esteganografía:** Es la robustez de la información oculta ante ataques intencionados.

- **Capacidad:** Mide la cantidad de información oculta que se puede incluir por cantidad de información portadora.

Así mismo también definiremos una serie de conceptos más específicos de este proyecto que posteriormente también vamos a utilizar:

- **Algoritmo**: Secuencia finita de instrucciones, reglas o pasos que describen de forma precisa las operaciones que un ordenador debe realizar para llevar a cabo una tarea en un tiempo finito. [Donald E. Knuth, 1968]

- **Puntos de esteganografiado**: Puntos del código donde es posible esteganografiar información, tales como las expresiones en sentencias condicionales "IF" y los bucles "WHILE" y "DO...WHILE".

- **Código cero**: Se entenderá por código cero el código fuente inicial en C que se usara como portador una vez que se ha trabajado sobre él para establecer a valor cero o valor inicial todos sus puntos de esteganografiado.

- **Par de esteganografiado**: Grupo de dos nombres de funciones o procedimientos donde es posible esteganografiar información variando la posición relativa de la función o el procedimiento dentro del código.

- **Par de parámetros**: Grupo de dos nombres de parámetros pertenecientes a un procedimiento o argumento donde es posible esteganografiar información variando su posición relativa en la cabecera.

- **Huella de esteganografiado:** Es una representación gráfica de los puntos de esteganografiado que posee un determinado código.

Se representa como una barra blanca horizontal (El fichero) que contiene una serie de líneas azules en su interior(Los puntos de esteganografiado) cuya separación y posición corresponde con las que tienen estos en el fichero de código.

1.3. OBJETIVOS DEL PROYECTO

En este proyecto se va a crear un programa escrito en lenguaje de programación "C" que sea capaz de esteganografiar información dentro en un código escrito también en lenguaje de programación "C". Para ello, se van a tener que realizar una serie de pasos.

- Estudiar las técnicas actuales de esteganografiado existentes para los distintos medios digitales para comprender su funcionamiento.

- Investigar el código escrito en lenguaje de programación "C" para averiguar donde se puede esteganografiar en el información, la capacidad de esteganografiado que se puede obtener y analizar su persistencia.

- Diseñar los distintos algoritmos de esteganografiado.

- Diseñar y codificar el programa en lenguaje de programación "C".

- Estegoanalizar los distintos métodos del programa para analizar su viabilidad y resistencia al ataque.

1.4. MOTIVACIÓN DEL PROYECTO

Este proyecto va a dar un nuevo uso a la esteganografía en informática cambiando al *portador.* Dejaremos de lado el normalmente usado fichero multimedia, para el cual existen actualmente multitud de herramientas tanto para su esteganografiado y desesteganografiado como para su estegoanálisis, para pasar a utilizar código escrito en el lenguaje de programación "C" como portador.

Esto nos llevara a poder obtener una serie de nuevos usos para la esteganografía que no se centran solo en el esteganografiado de información, sino que usa este mismo principio para una serie de aplicaciones prácticas.

- **ESTEGANOGRAFIADO DE INFORMACIÓN:**
  Se obtiene la capacidad de ocultar información dentro del propio código.

- **FIRMA DIGITAL DE UN CÓDIGO:**
  Se obtiene la capacidad de firmar el código de manera oculta al usuario final del código.

- **IDENTIFICACIÓN DE DISTINTAS COPIAS DE UN MISMO CÓDIGO:**
  Se obtiene la capacidad de asignar cada una de las copias de un mismo código a distintos usuarios finales con el fin de poder identificarlas unívocamente.

- **IDENTIFICACIÓN DE UN CÓDIGO DENTRO DE OTRO:**
  Se obtiene la capacidad de firmar el código de manera oculta y de poder ser este identificado cuando forma a su vez, parte de un código mayor.

Dichas funcionalidades se explican más adelante.

2. ESTADO DEL ARTE

2.1. ESTEGANOGRAFÍA EN LA HISTORIA

A continuación mostraremos cronológicamente como la esteganografía, presente desde tiempos antiguos, ha ido sufriendo una evolución constante hasta nuestros días:

Siglo V A.C. Comienzo de la esteganografía:

El historiador griego **Heródoto** nos describe en su libro "*Las Historias*" el método que tenían los griegos de enviarse mensajes esteganografiados entre sí usando las llamadas "*tablillas enceradas*".

> "Y por cuanto corría el peligro de ser interceptado el aviso, ni tenía otro medio para comunicárselo, valióse del siguiente artificio: Tomó un cuadernillo de dos hojas o tablillas; rayó bien la cera que las cubría, y en la madera misma grabó con letras la resolución del rey. Hecho esto, volvió a cubrir con cera regular las letras grabadas, para que el portador de un cuadernillo en blanco no fuera molestado de los guardas de los caminos."
>
> HERÓDOTO DE HALICARNASO.
> Los nueve libros de la historia Tomo VII.

Ilustración 1: Tablilla encerada con mensaje oculto grabado en la madera bajo la cera. Fuente: incibe.es

El mismo **Heródoto** narra también la historia de **Histieo,** el cual utiliza a un criado de su confianza como portador de un mensaje esteganografiado dirigido a **Aristagoras de Mileto**, solicitando su ayuda en una rebelión contra el rey de Persia.

> "Quiso a más de esto la casualidad que en aquella agitación le viniera desde Susa, de parte de Histieo, un enviado con la cabeza toda marcada con letras, que significaban a Aristagoras que se sublevase contra el rey. Pues como Histieo hubiese querido prevenir a su deudo que convenía rebelarse, y no hallando medio seguro para posarle el aviso por cuanto estaban los caminos tomados de parte del rey, en tal apuro había rasurado a navaja la cabeza del criado que tenía de mayor satisfacción, habíale marcado en ella con los puntos y letras que le pareció, esperó después que le volvieran a crecer el cabello, y crecido ya, habíalo despachado a Mileto sin más recado que decirle de palabra que puesto en Mileto pidiera de su parte a Aristagoras que, cortándole a navaja el pelo, le mirara la cabeza. Las notas grabadas en ella significaban a Aristagoras, como dije, que se levantase contra el Persa."
>
> HERÓDOTO DE HALICARNASO. Los nueve libros de la historia Tomo V.

Siglo XV. La esteganografía clásica:

El científico italiano **Giovanni Battista della Porta**, entre sus muchas actividades, se dedicó también a la criptografía. En 1563, publico su libro "*De Furtivis Literarum Notis, vulgo de ziferis*"[4], donde expuso los conocimientos criptográficos de la época.

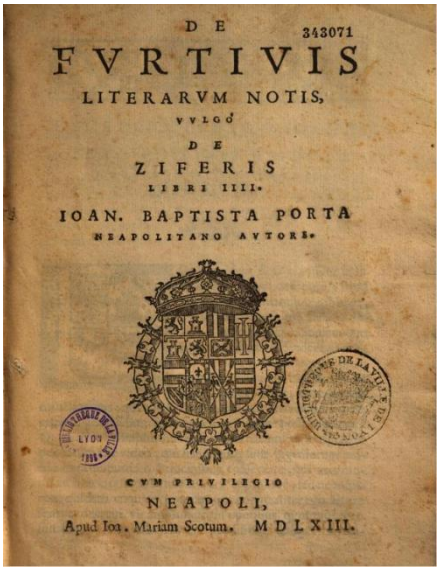
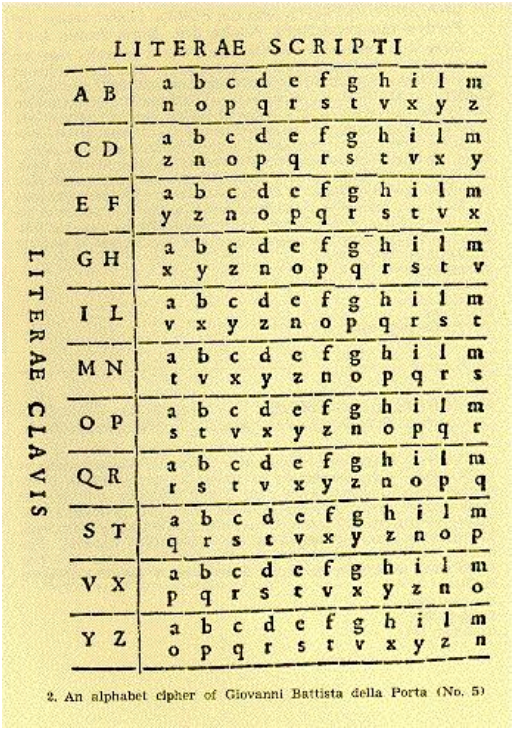


Ilustración 2: Portada del libro "*De Furtivis Literarum Notis*" Fuente: https://books.google.es.

De su obra cabe destacar principalmente, la primera técnica de cifrado por sustitución digráfica, donde son sustituidas cada par de letras por un símbolo diferente, el sistema de cifrado por sustitución polialfabética, y la técnica de ataque por palabra probable, que consiste en deducir la clave del mensaje suponiendo la existencia de una palabra probable en el mensaje cifrado.



2. An alphabet cipher of Giovanni Battista della Porta (No. 5)

Ilustración 3: Alfabeto de cifrado del libro "De Furtivis Literarum Notis" Fuente: https://books.google.es.

Della Porta empleó 11 alfabetos diferentes y reversibles que él designó por AB, CD, EF, etc.

El proceso de cifrado es muy sencillo, para cifrar la cadena NOELIA por ejemplo, primero se escoge un alfabeto, Alfabeto AB. Ahora, para cifrar cada letra, se busca está en la fila del alfabeto seleccionado y se cambia por la letra que se encuentre abajo o arriba de esta, dependiendo de en qué fila se encuentre.

N = a, O = b, E = r, L = y, I = x, A = n.
La cadena NOELIA cifrada seria "abryxn".

Se puede ver que este cifrado es reversible y que si volvemos a cifrar el texto cifrado con el mismo alfabeto obtendremos de nuevo el texto en claro original.

También describe con todo lujo de detalles la manera de esconder un mensaje dentro de un huevo cocido. Técnica que utilizaba para comunicarse con algunos de sus amigos presos por la Inquisición debido a su afición compartida por el ocultismo, debido a que en aquella época era habitual llevar alimentos a los presos.

El método consistía en preparar una tinta mezclando una onza de alumbre con una pinta de vinagre, y luego, escribiendo el mensaje en la cáscara del huevo crudo. Como la cáscara del huevo es porosa, la tinta penetraba hasta el interior y dejaba el mensaje en la superficie de la albúmina del huevo. Cuando la tinta estaba seca, se cocía el huevo, de manera que la tinta que quedaba en el exterior se borraba, por lo que el mensaje sólo podía ser leído si se pelaba el huevo, donde el mensaje aparecía impreso en la clara.

[4] Libo disponible para su consulta online en https://books.google.es/books?id=sc-Zaq8_jFIC

Otro ejemplo histórico de la esteganografía de ese periodo es el libro de **Francesco Colonna** "*Hypnerotomachia Poliphili"*. En él, se puede apreciar la propia firma del autor en un acróstico esteganografiado en el propio libro. Si se toma la primera letra de sus 38 capítulos se puede leer "*Poliam frater Franciscus Columna peramavit*", que se traduce por "*El hermano Francesco Colonna ama apasionadamente a Polia*".

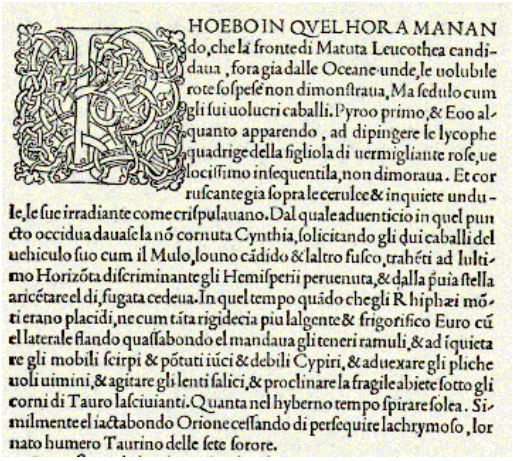

HOEBO IN QVEL HORA MANAN
do, che la fronte di Matuta Leucothea candi-
daua, fora gia dalle Oceane unde, le uolubile
rote ſoſpeſe non dimonſtraua, Ma ſedulo cum
gli ſui uolucri caballi. Pyroo primo, & Eoo al-
quanto apparendo, ad dipingere le lycophe
quadrige della figliola di uermigliante roſe, ue
locisſimo inſequentila, non dimoraua. Et cor
ruſcante gia ſopra le cerulee & inquiete undu-
le, le ſue irradiante come criſpulauano. Dal quale aduenticio in quel pun
cto occidua dauaſe la nō cornuta Cynthia, ſolicitando gli dui caballi del
uehiculo ſuo cum il Mulo, louno cādido & laltro fuſco, trahēti ad Iulti-
mo Horizōta diſcriminante gli Hemiſperii peruenuta, & dalla puia ſtella
aricētare el di, fugata cedeua. In quel tempo quādo chegli Rhiphæi mō-
ti erano placidi, ne cum tāta rigidecia piu lalgente & frigorifico Euro cū
el laterale flando quaſſabondo el mandaua gli teneri ramuli, & ad ſquieta
re gli mobili ſcirpi & pōtuti iūci & debili Cypiri, & aduexare gli pliche
uoli uimini, & agitare gli lenti ſalici, & proclinare la fragile abiete ſotto gli
corni di Tauro laſciuianti. Quanta nel hyberno tempo ſpirare ſolea. Si-
milmente el iactabondo Orione ceſſando di perſequire lachrymoſo, lor
nato humero Taurino delle ſete ſorore.

Ilustración 5: Fragmento del primer capítulo de "Hypnerotomachia Poliphili"

Ilustración 4: Vista en detalle de la letra capitular del primer capitulo de "Hypnerotomachia Poliphili"

Agrupación de las 38 letras capitulares del libro *"Hypnerotomachia Poliphili"* donde se aprecia el acróstico "*Poliam frater Franciscus Columna peramavit*"*:*

Ilustración 6: P O L I A M

Ilustración 7: F R A T E R

Ilustración 10: F R A N C I S C V S

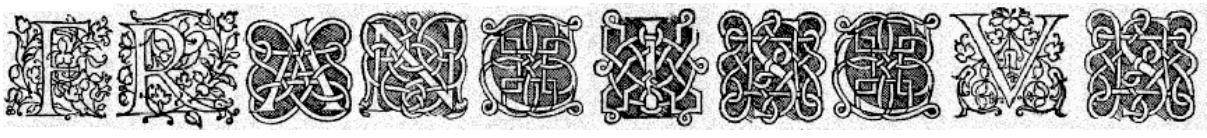


Ilustración 9: C O L U M N A

Ilustración 8: P E R A M A V I T

Fuente de las Imágenes: mitpress.mit.edu

Siglo XVI. La esteganografía maldita:

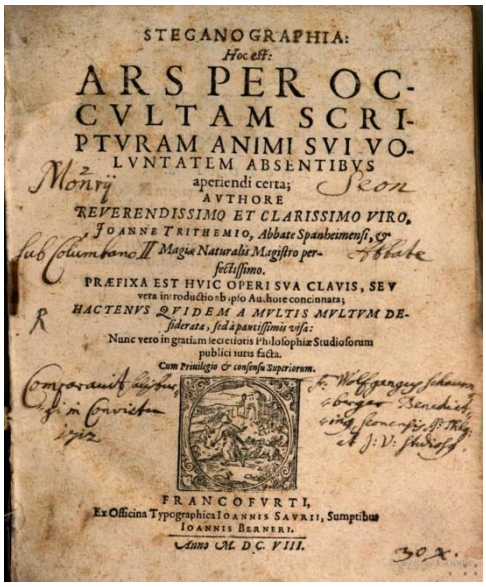

STEGANOGRAPHIA:
Hoc eſt:
ARS PER OC-
CVLTAM SCRI-
PTVRAM ANIMI SVI VO-
LVNTATEM ABSENTIBVS
aperiendi certa;
AVTHORE
REVERENDISSIMO ET CLARISSIMO VIRO,
IOANNE TRITHEMIO, Abbate Spanheimenſi, &
Columbano II Magiæ Naturalis Magiſtro per-
fectiſſimo.
PRÆFIXA EST HVIC OPERI SVA CLAVIS, SEV
vera introductio ab ipſo Authore concinnata;
HACTENVS QVIDEM A MVLTIS MVLTVM DE-
ſiderata, ſed à pauciſſimis viſa:
Nunc vero in gratiam ſecretioris Philoſophiæ Studioſorum
publici iuris facta.
Cum Priuilegio & conſenſu Superiorum.
FRANCOFVRTI,
Ex Officina Typographica IOANNIS SAVRII, Sumptibus
IOANNIS BERNERI.
Anno M. D. C. VIII.

Ilustración 11: Portada de "Steganographia" Fuente: biblioteca Archive.org.

El vocablo esteganografía apareció por primera vez cuando el abad alemán **Johannes Trithemius** escribió los tres volúmenes de su libro "*Steganographia*"[5]. En él, explica cómo enviar mensajes en menos de un día y sin usar ningún medio físico, como pueden ser cartas, libros o mensajeros. Sino mediante espíritus con la llamada magia angélica. De los tres libros de la *Steganographia*, en los volúmenes I y II, Trithemius nos cuentan la manera de conseguir la ayuda de los espíritus.

El método a estas alturas del documento y con el conocimiento que poseemos sobre esteganografía nos resultara conocido: se escribe la carta, se recita el ritual de invocación apropiado y el espíritu cartero aparecerá para llevar el mensaje al destinatario, que lo recibirá si usa a la vez, la invocación oportuna. Es curioso que este método necesite un conjuro para enviar y a su vez un conjuro oportuno para recibir, vamos, lo que viene conociéndose como una clave.

Uno de estos conjuros comenzaba por la siguiente frase:

"Padiel aporsy mesarpon omeuas peludyn malpreaxo..."

Que tras aplicar el código de transposición de letras con una secuencia determinada que utilizó Trithemius, el resultado es:

*"padiel a**PoRsY** mesarpon o**MeUaS** peludyn m**AlPrEaX**o..."*

La solución es la expresión latina "*PRIMUS ÁPEX*", "La primera cumbre" en latín.

Por lo tanto, se demuestra que Trithemius había descubierto una forma de enviar mensajes secretos a distancia.

Por el contrario, en el volumen III, ya no es fundamental la invocación, sino el cálculo astrológico correspondiente al ángel y la hora en que se realiza. Debido a que como se descubrió más adelante, Trithemius utilizo la técnica del alfabeto inverso para esteganografiar sus mensajes.

Aunque todos los mensajes esteganografiados de los tres volúmenes resultaron ser frases, aparentemente escogidas al azar, no es de extrañar que semejante texto entrase de cabeza en el Índice de los Libros Prohibidos de la Iglesia. Debido a que la Iglesia le acusó de enseñar las ciencias malditas y de hacer sortilegios diabólicos.

El libro es considerado como el más influyente tanto en la evolución de la criptografía como en la historia de la esteganografía propiamente dicha, ya que define sistemas para ocultar mensajes secretos dentro de mensajes aparentemente inofensivos y demuestra que Johannes Trithemius es, sin lugar a dudas, la figura más destacada de la moderna ciencia de la criptografía.

Aparte de este "*Steganographia*", también publicó "*Polygraphiae Libri Sex*", una colección de seis libros sobre criptografía que no contenían los elementos esotéricos de su otro gran libro.

[5] Libo disponible para su consulta online en https://archive.org/details/SteganographiaBSB1608

Siglo XX. Esteganografía moderna:

Durante la Segunda Guerra Mundial la esteganografía sufrió un aumento significativo en su utilización siendo utilizado por todos los bandos y en todas las posibles situaciones.

Uno de los métodos más utilizados era el de microfilmar un mensaje y reducirlo hasta el extremo de un minúsculo punto, el cual, podía pasar como un signo de puntuación de un carácter dentro de otro mensaje.

Ilustración 12: Ejemplo de micropunto. ***Fuente:*** **elreservado.es**

Otro método menos sofisticado era el usado por los prisioneros, los cuales usaban los puntos de la i y la j y las rayas de la t y la f para enviarse mensajes esteganografiados en código morse entre ellos.

De forma similar, también se hacían pequeños orificios en determinadas letras de un periódico para que al sostenerlo frente a una fuente de luz se pudieran observar todas aquellas letras marcadas y de esta forma obtener el mensaje esteganografiado.

En la esteganografía también se incluye la práctica de escribir con tinta invisible, método utilizado por casi todas las culturas.

Una técnica básica consiste en utilizar sustancias que contienen goma, mucílago, albúmina, azúcar o un alto contenido en carbono, tales como la leche, el zumo de algunas frutas, el vino o el vinagre entre otros, para dejar un mensaje oculto que solo será revelado cuando se caliente la superficie donde se escribió.

2.2. TÉCNICAS DIGITALES DE ESTEGANOGRAFIADO

Actualmente, la esteganografía, se ha vuelto digital debido a que la informática y los ordenadores, permiten no sólo ocultar mensajes, sino, ficheros multimedia completos unos dentro de otros. Existen numerosos métodos y algoritmos que posibilitan el esteganografiado de archivos dentro de archivos multimedia (imágenes, audio y vídeo) aparentemente inofensivos. A continuación se indican algunos de los más usados.

Enmascaramiento y filtrado (Masking and Filtering)

Esta técnica consiste en ocultar dentro de una imagen digital una serie de información complementaria a esta como pueden ser el copyright, los datos del autor o la licencia.

Esto, difiere de la esteganografía tradicional, donde básicamente se trata de transmitir información encubierta, debido a que su finalidad es ampliar la cantidad de información presente en una imagen digital al añadir a la imagen que actúa como portador una serie de información para que en caso necesario pueda demostrase el uso fraudulento o ilícito de dicha imagen.

Método de la marca de agua digital: Se trata de un identificador digital persistente e imperceptible agregado a las imágenes para comunicar la titularidad de los derechos de autor y ayudar a su localización cuando se utilice online. Aunque una marca de agua digital incrustada generalmente es imperceptible para el ojo humano, pero pueden ser detectadas por un lector digital de marcas de agua, también existen marcas de agua que son perceptibles a simple vista. Las típicas marcas de agua superpuestas a una imagen y que sirven para indicar claramente la propiedad de la misma.

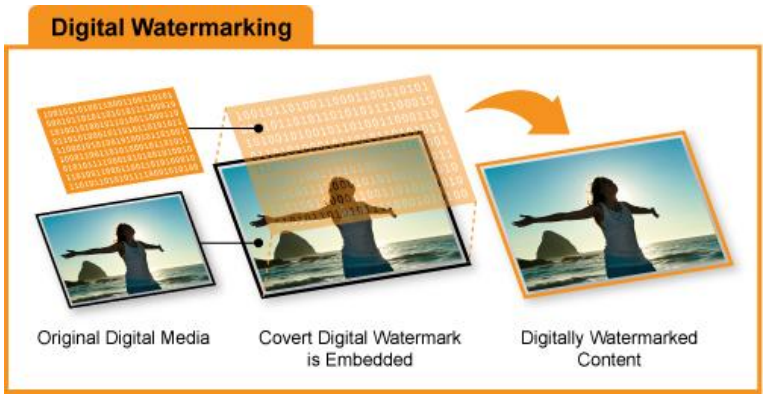


Ilustración 13: Ejemplo de marca de agua digital. ***Fuente*****: licensestream.com**

Método del fingerprinting o huella digital: Este método analiza el contenido de la imagen y luego crea una huella digital única de la imagen donde introduce no sólo información sobre el autor sino además información del usuario que ha adquirido los derechos de uso de la misma. De este modo se puede utilizar este identificador único de la imagen para su seguimiento a través de la Web y perseguir la distribución ilegal de este tipo de servicios digitales.

Ilustración 14: Ejemplo de huella digital. ***Fuente*****: licensestream.com**

Algoritmos y transformaciones (Algorithms and Transformations)

Este método consiste en ocultar información en los bits de datos menos significativos del objeto basándose en las funciones matemáticas utilizadas en los algoritmos de compresión y transformación de datos.

Ocultación en el bit menos significativo (LSB: Least Significant Bit)

Este método es uno de los llamados métodos de sustitución y consiste en sustituir el bit menos significativo, el último bit de cada byte, de los pixeles de una imagen por los bits de la información a ocultar. Este proceso puede repetirse con cada byte de la imagen esparciendo el mensaje por toda la imagen sin que se aprecie diferencia alguna a simple vista.

Este método funciona mejor cuanto mayor tamaño y más variaciones y profundidad de color posea la imagen. Debido a que esto proporciona mayor capacidad y mejor transparencia de los bits modificados.

Este mismo método también puede ser aplicado a ficheros de vídeo y audio siguiendo el mismo proceso y utilizando los cambios de frecuencias o los bits inaudibles que pueden ser reemplazados por los bits del mensaje a ocultar.

Este método se explica más detalladamente en el apartado 2.3 Esteganografiado en imágenes.

Inserción de bits en el objeto contenedor

El método de inserción de bits implica encajar el mensaje secreto directamente en el objeto portador a partir de una determinada parte del fichero (fin de fichero, espacios de alineamiento, metadatos de cabecera, etc.).

El principal inconveniente que tiene este método frente a otros reside en que esta inserción de bits hace que el fichero resultante aumente de tamaño frente al fichero original.

En una imagen BMP los primeros 54 bytes corresponden a los metadatos de la imagen y su estructura es la siguiente:

Ilustración 15: Esquema de una imagen BMP.

2 bytes – Tipo de fichero.
4 bytes – Tamaño del archivo.
4 bytes – Reservados para usos futuros.
4 bytes – Offset, distancia entre la cabecera y los píxels de la imagen
4 bytes – Tamaño de los Metadatos.
4 bytes – Ancho de la imagen.
4 bytes – Alto de la imagen.
2 bytes – Numero de planos de color.
2 bytes – Profundidad de color.
4 bytes – Tipo de compresión.
4 bytes – Tamaño de la estructura de la imagen.
4 bytes – Píxeles por metro horizontal.
4 bytes – Píxeles por metro vertical.
4 bytes – Cantidad de colores usados.
4 bytes – Cantidad de colores importantes.

Por lo tanto, la manera más fácil de ocultar información en una imagen BMP consiste en introducirla entre los metadatos y los datos de la imagen, modificando a su vez el tamaño del campo offset para dejar hueco suficiente a todo el contenido adicional que se le va a añadir.

Este método se ve más claro con un ejemplo:

Tomando como imagen BMP el logo de la universidad, se puede observar en el Byte 10 de sus metadatos como el tamaño del offset en Hexadecimal es de 36 Bytes (posición hexadecimal A-00h) o lo que es lo mismo 54 Bytes en decimal, lo cual nos indica que el primer pixel de la imagen está situado en el Byte 54 (posición hexadecimal 36h).

Ilustración 16: Logo de la Universidad Carlos III de Madrid
Fuente: Uc3m.es

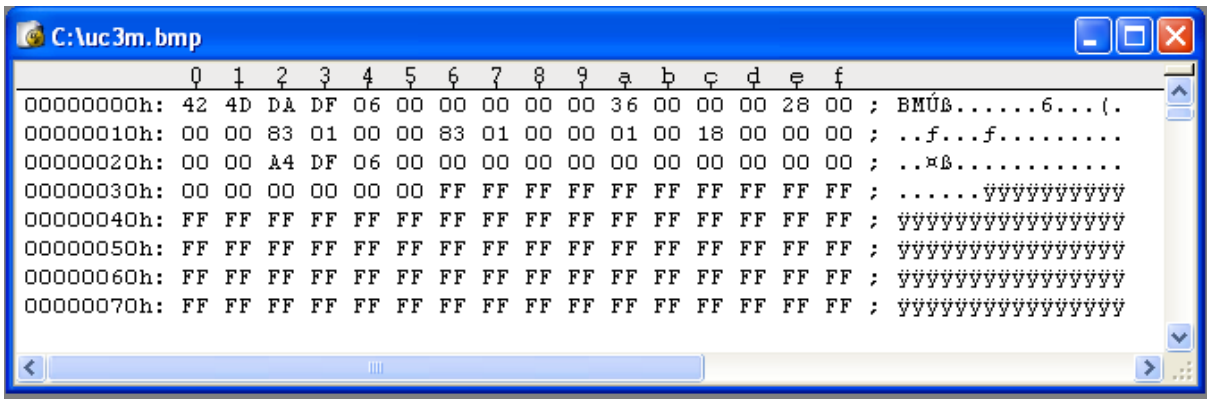


Ilustración 17: Fichero de Bytes de la imagen BMP sin esteganografiar

Si variamos el tamaño del offset sumándole el tamaño del mensaje a esteganografiar, este se puede ocultar sin problemas en la imagen debido a que los Bytes que forman parte de la imagen no se han modificado. El Byte 10 muestra el nuevo valor del offset el cual ahora es de 88 Bytes (58 Hex) que corresponden a los 54 iniciales más los 34 del mensaje a esteganografiar.

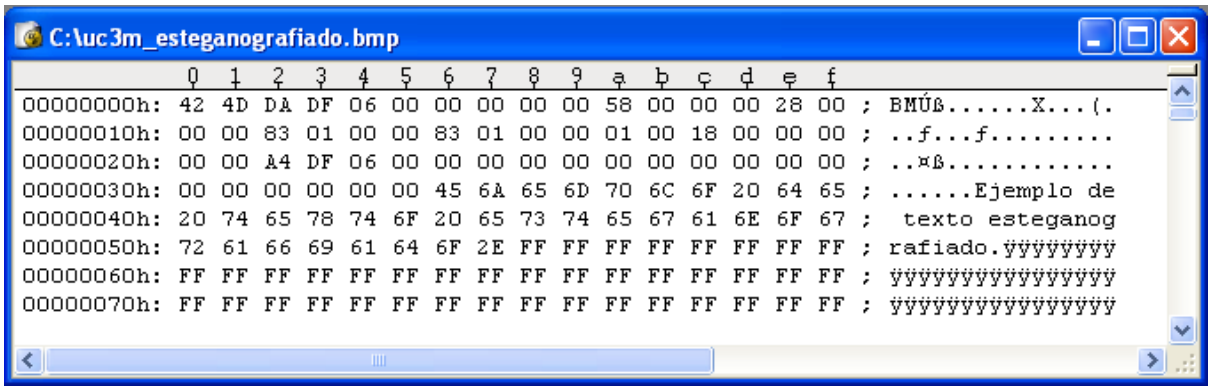


Ilustración 18: Fichero de Bytes de la imagen BMP esteganografiada

Creación de un fichero contenedor propio partiendo de la información a ocultar

Este método consiste en crear un fichero contenedor desde cero partiendo de la propia información a ocultar. La ventaja fundamental de este método es que no existe un fichero original con el que comparar el fichero esteganografiado para buscar posibles diferencias.

Un ejemplo de este método lo podemos observar en la web "*spammimic.com*" donde por medio de un algoritmo de creación esteganográfico y partiendo de un mensaje del usuario, se crea un correo de *spam* donde está oculto dicho mensaje. La propia web *spammimic.com* nos dice el porqué de utilizar este método y en particular su aplicación:

> "There are terrific tools (like PGP and GPG) for encrypting your mail. If somebody along the way looks at the mail they can't understand it. But they do know you are sending encrypted mail to your pal.
>
> The answer: encode your message into something innocent looking.
>
> Your messages will be safe and nobody will know they're encrypted!
>
> There is tons of spam flying around the Internet. Most people can't delete it fast enough. It's virtually invisible. This site gives you access to a program that will encrypt a short message into spam. Basically, the sentences it outputs vary depending on the message you are encoding. Real spam is so stupidly written it's sometimes hard to tell the machine written spam from the genuine article."
>
> **Fuente: *spammimic.com.***

Ilustración 19: Ejemplo de utilización de la web "spammimic.com"

2.3. TÉCNICAS MÁS UTILIZADAS SEGÚN EL TIPO DE MEDIO

Esteganografiado en imágenes

Si partimos de que para un ordenador, una imagen no es más que un array de números que muestran los diferentes colores e intensidades de luz que tiene cada pixel. El método más eficaz para el esteganografiado en imágenes es el método **LSB** (del inglés Least Significant Bit o Bit menos significativo), en el, para cada uno de los pixel que forman la imagen, se utiliza su bit menos significativo para almacenar información, y así obtener una imagen prácticamente igual a la original pero con un mensaje oculto en ella.

Cuanto de más alta calidad y mayor resolución es una imagen, más fácil y eficiente resulta esteganografiar la información dentro de ella. Pero cabe destacar que este método de esteganografiado no resulta persistente debido a que si el archivo de imagen es convertido a otro formato, la información que en él estaba esteganografiada puede resultar alterada o incluso eliminada.

En el siguiente apartado se puede comprobar como los cambios realizados en la imagen portadora una vez esteganografiada resultan prácticamente imperceptibles.

Ejemplo del método LSB para esteganografiado en imágenes:

Como ya se comentó anteriormente, este método consiste en sustituir el bit menos significativo, el último bit de cada byte, de los pixeles de una imagen por los bits de la información a ocultar.

ASCII	Decimal	Binario	Bit menos significativo
U	85	01010101	01010101
C	67	01000011	01000011
3	51	00110011	00110011
M	77	01001101	01001101

Tabla 1: Bit menos significativo caracteres "UC3M"

En la tabla anterior podemos observar cual es el bit menos significativo de los caracteres “UC3M”. Ahora, lo que haremos, será ocultar en el bit menos significativo de cada pixel del logo de la universidad un mensaje. Utilizaremos una imagen PNG puesto que este formato permite almacenar imágenes con una mayor profundidad de contraste y usa tres bytes para definir cada color.

Ilustración 20: Logo PNG de la Universidad Carlos III de Madrid Fuente: Uc3m.es

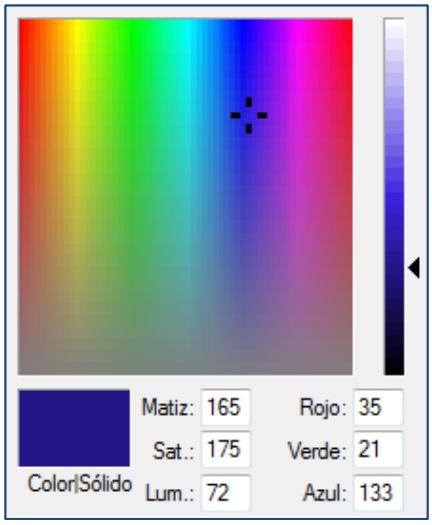


Ilustración 21: Análisis RGB del color de un pixel del logo de la universidad.

En la ilustración 21 podemos observar como el color de un pixel del logo está formado por tres bytes de colores, el rojo, el verde y el azul, lo que se conoce como color RGB por sus siglas en ingles de “Red Green Blue”.

En la siguiente tabla podemos observar cual es el bit menos significativo de cada uno de los bytes que forman el color del pixel.

Byte	Decimal	Binario	Bit menos significativo
Rojo	35	00100011	00100011
Verde	21	00010101	00010101
Azul	133	10000101	10000101

Tabla 2: Bit menos significativo del pixel del logo.

El método de esteganografiado LSB, ahora consiste en ocultar nuestro mensaje en binario en los bits menos significativos de cada pixel de la imagen. Como para ocultar nuestro mensaje “UC3M” que en binario seria 01010101010000110011001101001101 necesitaríamos 11 pixel. Para no repetir 11 veces la misma explicación, vamos modificar solo un pixel a modo de ejemplo.

Byte	Decimal Original	Binario Original	Binario Modificado	Decimal Modificado
Rojo	35	00100011	00100010	34
Verde	21	00010101	00010101	21
Azul	133	10000101	10000100	132

Tabla 3: Bit modificado del pixel del logo.

Las siguientes ilustraciones muestran el resultado de la modificación del pixel anterior, donde no se aprecia prácticamente ningún cambio:

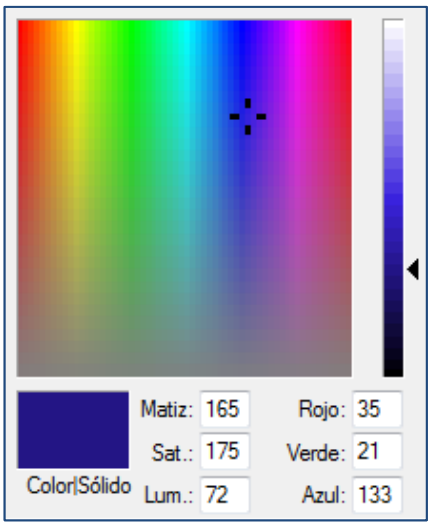


Ilustración 22: Análisis RGB del pixel original.

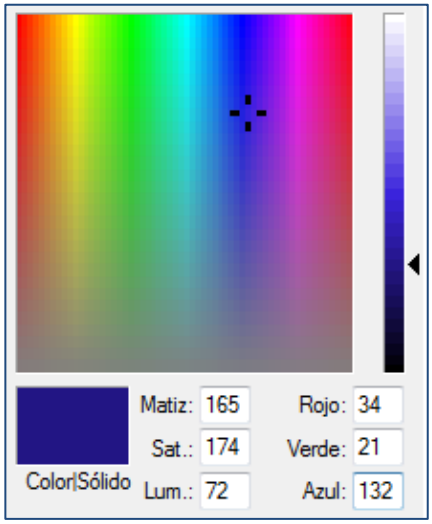


Ilustración 23: Análisis RGB del pixel modificado.

En el resultado del esteganografiado total de la imagen, mostrado en las dos siguientes ilustraciones, se puede ver que el cambio de color también es prácticamente inapreciable:

Ilustración 24: Logo PNG original

Ilustración 25: Logo PNG modificado

Esteganografiado en audio

El esteganografiado en este medio se basa principalmente en explotar la incapacidad que tiene el oído humano para detectar una serie de cambios dentro de una misma frecuencia.

Los métodos más utilizados para la esteganografiado de información en ficheros de audios son las siguientes:

- **Bit menos significativo:** Es similar al método utilizado en las imágenes donde para cada uno de los bytes que forman el audio, se utiliza su bit menos significativo para almacenar información en él. Las modificaciones que realiza este método aunque son prácticamente imperceptible para el ojo humano en imágenes, generalmente si son perceptibles por el oído cuando se producen en el audio portador, por lo que no resulta un método muy eficaz.

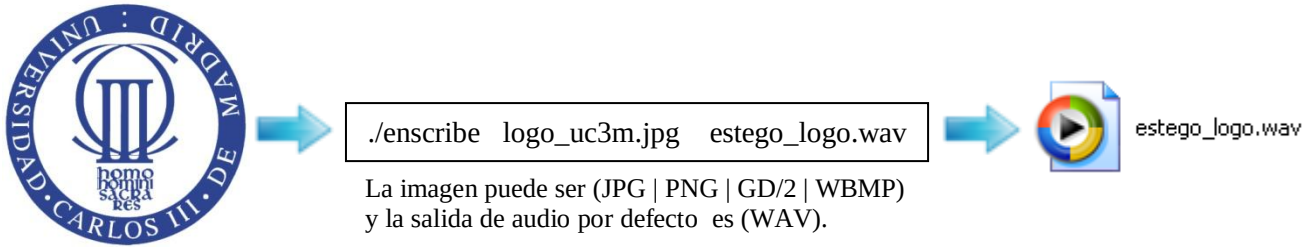
- **Espectro ensanchado:** Este método consiste en ocultar un mensaje en una determinada frecuencia dentro de otro con una frecuencia mayor, añadiendo a su vez una serie de ruido para mejorar la ocultación.

- **Ocultación en ecos**: Este método consiste en ocultar el mensaje secreto añadiéndolo en el eco del audio portador.

- **Máscara perceptual**: Este método se basa en ocultar el audio del mensaje tras otro sonido que posea la misma frecuencia a modo de mascara para que el primero pare inadvertido.

Ejemplo de esteganografiado de una imagen en un fichero de audio:

Para este ejemplo usaremos un software-libre llamado "*enscribe*" [6] desarrollado por **Jason Downer** para ocultar la imagen del logo de la universidad en un archivo de audio WAV.



La imagen puede ser (JPG | PNG | GD/2 | WBMP)
y la salida de audio por defecto es (WAV).

El programa nos devuelve como salida en este caso un archivo de audio WAV de 11 segundos de duración en el que aparentemente no hay nada más que ruido, pero lo que nos interesa no es el sonido sino el espectrograma de dicho sonido.

[6] http://coppercloudmusic.com/enscribe/

Si abrimos el fichero de audio con un programa para ver espectrogramas como puede ser "Baudline" [7] lo que nos aparece es esto:

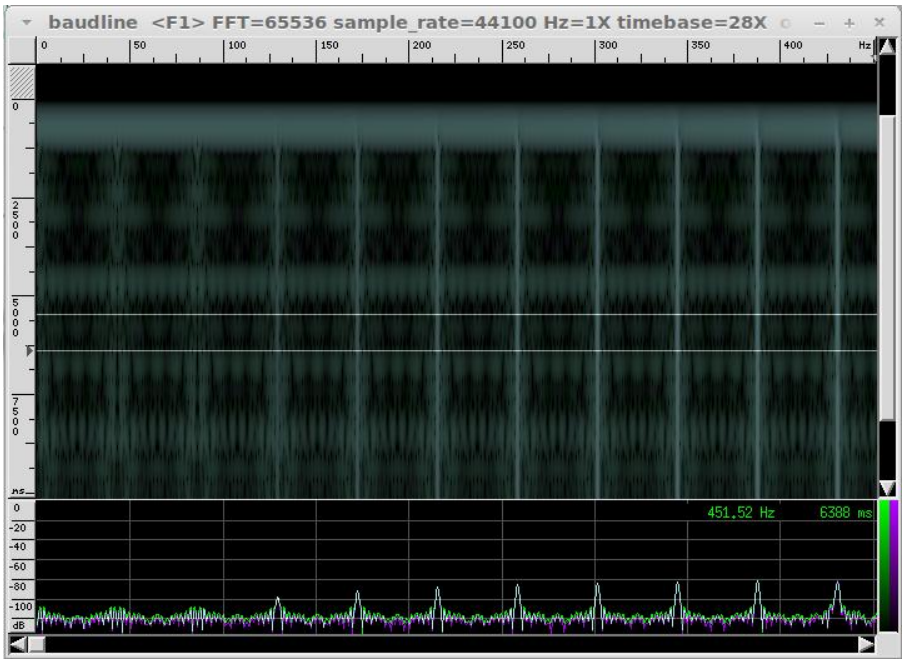


Ilustración 26: Visualización normal del espectrograma de audio

Pero si a ese mismo espectrograma le quitamos zoom y lo vemos todo a la vez por pantalla lo que aparece es una cosa muy distinta, puesto que las líneas del espectrograma agrupadas muestran nuestra imagen esteganografiada:

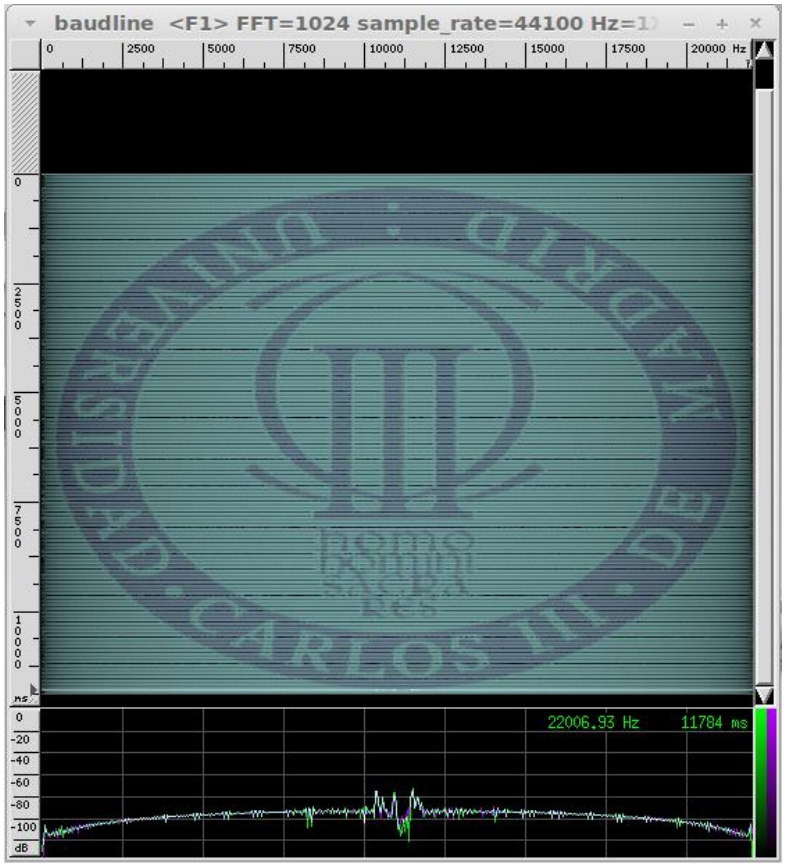


Ilustración 27: Visualización del espectrograma de audio sin zoom

[7] http://www.baudline.com/

Esteganografiado en documentos

La esteganografía en los documentos puede dividirse en dos grandes tipos dependiendo de si se añade información o por el contrario se modifica la información del propio documento portador.

- **Método de inserción de blancos.**

Este método de esteganografía sobre texto consiste en ocultar los bits del mensaje insertando espacios en blanco en el texto original. Estos espacios en blanco se insertan tanto entre las palabras como al final de las líneas.

- **Métodos de modificación:**

Estos métodos se basan en la modificación del propio texto del documento, tanto sintácticamente como semánticamente para ocultar en dichos cambios el mensaje a esteganografiar.

La modificación sintáctica consiste en la modificación de los signos de puntuación y del orden de algunas frases y la semántica en la utilización de los sinónimos de algunas palabras clave del texto.

Ejemplo de modificación sintáctica:

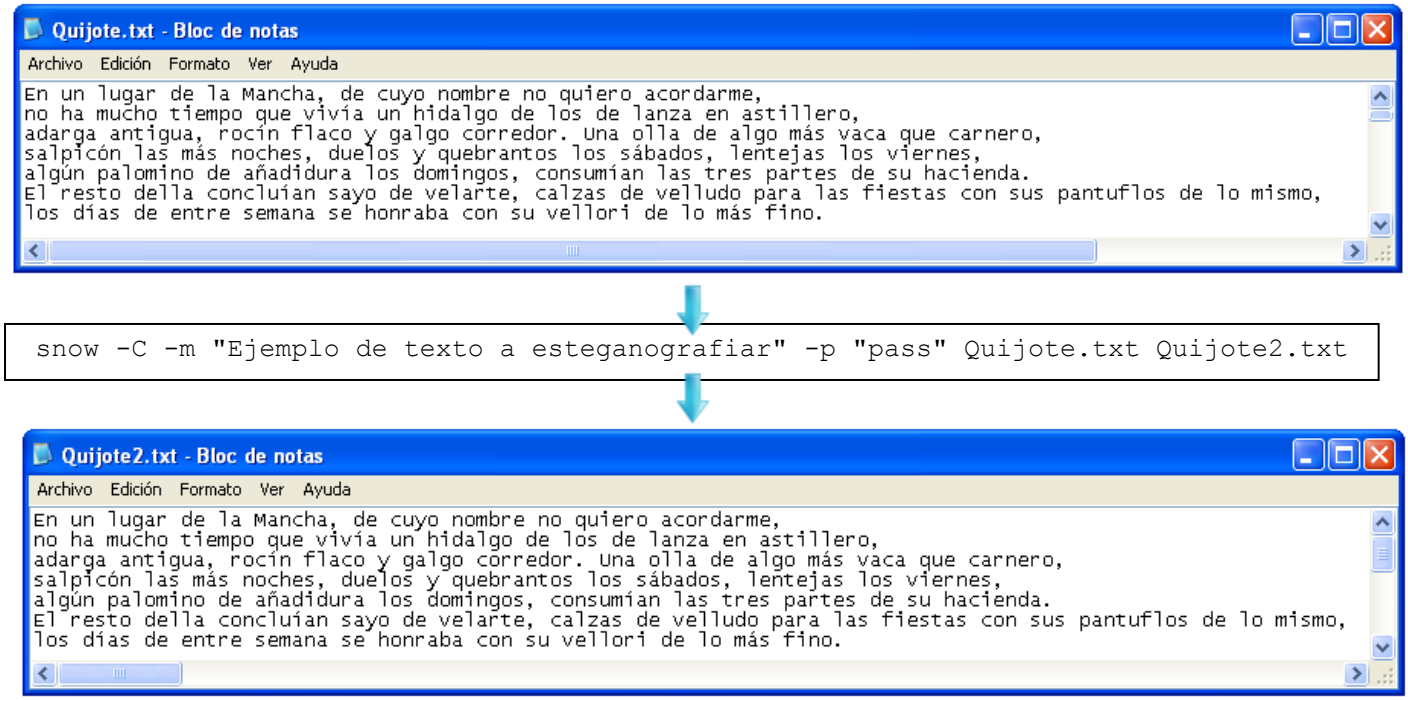
Original: Volveré mañana por la mañana. → Modificada: Mañana volveré por la mañana

Ejemplo de modificación semántica:

Original: Volveré pronto. → Modificada: Volveré temprano.

Ejemplo de esteganografiado de información por inserción de blancos en un fichero de texto:

Para este ejemplo usaremos un software-libre llamado "snow" [8] desarrollado por **Matthew Kwan** para ocultar una cadena de información cifrada en un archivo de texto.



[8] http://www.darkside.com.au/snow/

Ejecutamos el programa *"Snow"* pasándole como argumentos el mensaje a cifrar: `Ejemplo de texto a esteganografiar`, la contraseña para cifrar: `pass` y los ficheros de entrada y de salida. Aparentemente el programa nos devuelve un fichero idéntico al de entrada, tanto es así, que incluso algunos programas de comparación no son capaces de encontrar las diferencias:

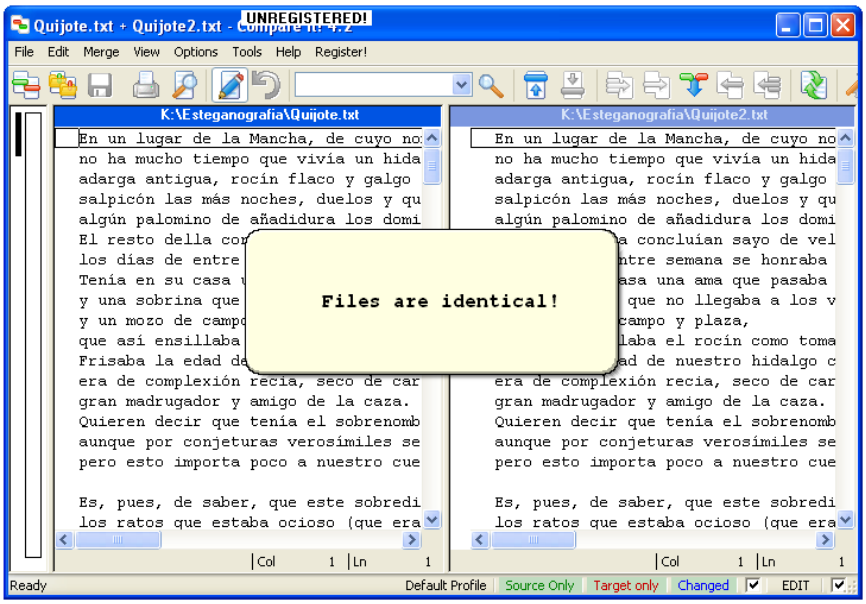


Ilustración 28: Resultado de comparar los ficheros con el programa "Compare It!"

Diferencias que si se pueden encontrar al conocer el método de esteganografiado fijándonos en la longitud de las frases incluyendo los espacios en blanco:

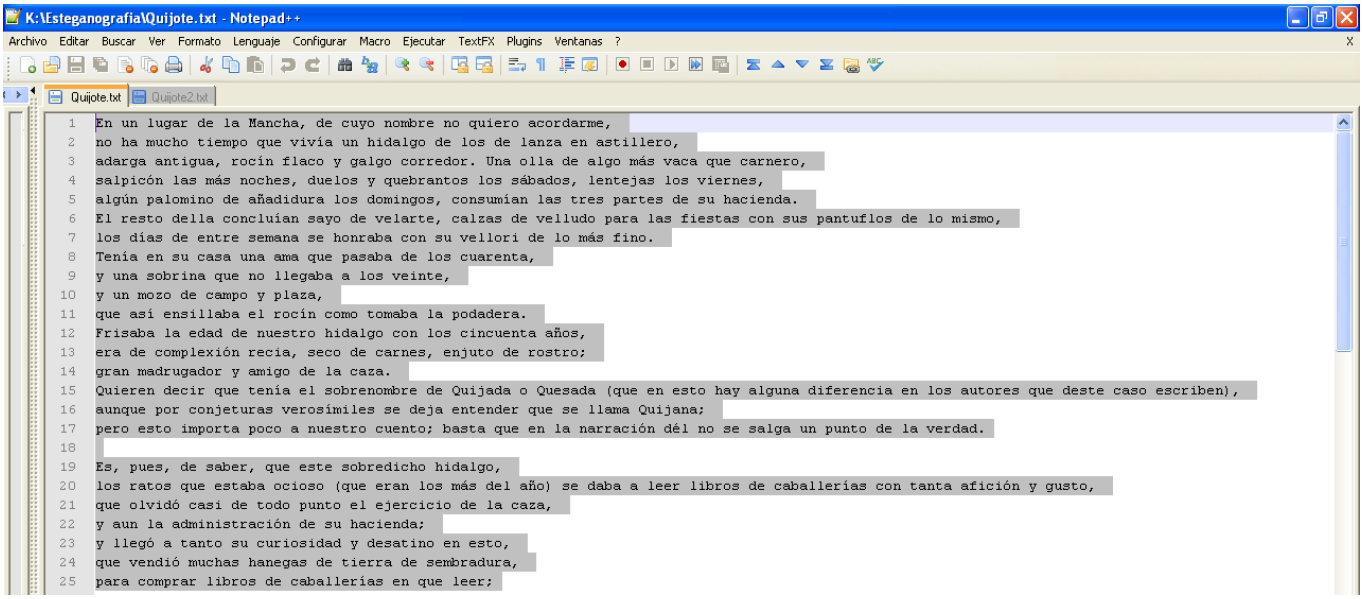


Ilustración 29: Fichero de texto del Quijote sin esteganografía

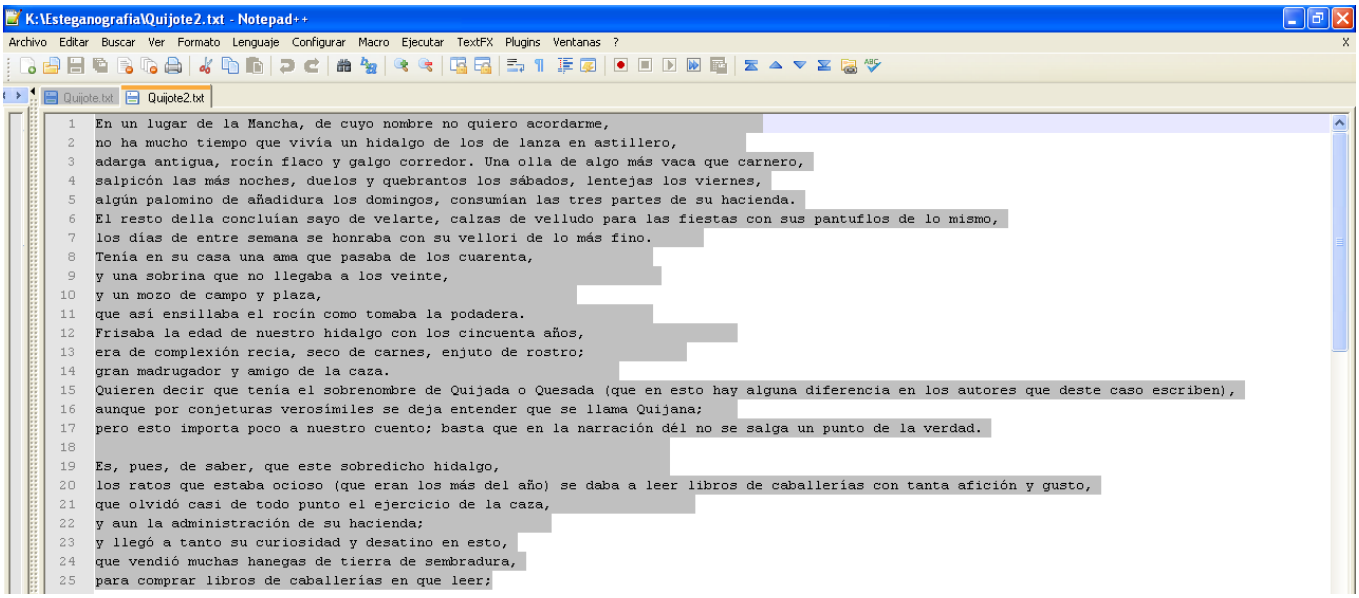


Ilustración 30: Fichero de texto del Quijote con esteganografía

Esteganografiado en videos

Cuando queremos ocultar información en vídeos se suele utilizar el llamado método DCT (Transformada de coseno discreta) basado en la transformada discreta de Fourier, pero utilizando solo números reales.

El método DCT funciona como los métodos de esteganografiado en imágenes pero aplicados a cada uno de los fotogramas del video. El procedimiento aplicando es el siguiente:

1. Calcular la transformada de coseno discreta de la imagen.
2. Sustituir los coeficientes menores a un valor umbral por los bits a ocultar.
3. Calcular la inversa de la transformada de coseno discreta de la imagen.
4. Almacenar.

Otras técnicas de esteganografía

En las redes TCP/IP se está implementando una nueva técnica esteganográfica basada en incluir a los paquetes enviados por la red una serie de retardos o "*delays*" en los cuales puede ser esteganografiada información.

2.4. HERRAMIENTAS PARA ESTEGANOGRAFÍA

La siguiente tabla muestra algunas de las herramientas para esteganografía que existen actualmente, así como el tipo de mensaje que pueden ocultar y el tipo de estego-objeto que utilizan:

Nombre	Medio	Tipo objeto mensaje	Tipo objeto portador
MP3Steqo[9]	Audio	Fichero de texto	Audio MP3
JPHide y JPSeek[10]	Imagen	Fichero de texto	Imagen JPEG
BlindSide Cryptographic Tool[11]	Imagen	Múltiples formatos	Imagen BMP
G1FShufffe[12]	Imagen	Fichero de texto	Imagen GIF
wbStego[13]	Imagen y fichero de texto	Múltiples formatos	Imagen BMP y fichero de texto, HTML o PDF
StreamSteganography[14]	Imagen	Fichero de texto	Imagen PNG
StegoVideo[15]	Video	Múltiples formatos	Video AVI

Tabla 4: Herramientas de esteganografía.

[9] **MP3Steqo**: http://www.petitcolas.net/fabien/steganography/mp3stego/
[10] **JPHide y JPSeek:** http://linux01.gwdg.de/~alatham/stego.html
[11] **BlindSide Cryptographic Tool:** http://lwww.mirrors.wiretapped.net/security/steganography/blindside/
[12] **G1FShufffe:** http://www.darkside.com.au/gifshuffle/
[13] **wbStego:** http://wbstego.wbailer.com/
[14] **StreamSteganography:** http://www.phpclasses.org/package/6027-PHP-Store-and-hidden-information-in-PNG-images.html
[15] **StegoVideo:** http://compression.ru/video/stego_video/index_en.html

2.5. ESTEGOANÁLISIS

Estegoanálisis es la técnica mediante la cual se intenta detectar un mensaje esteganografiado dentro de un medio. Tendremos dos tipos de ataques: ataques pasivos y ataques activos. A continuación se detallan ambos métodos.

Ataque activo

En un ataque activo el uso de la esteganografía ya es conocido y su finalidad simplemente es modificar el estego-objeto original para destruir la información en el esteganografiada. Este ataque se utiliza principalmente contra las técnicas de esteganografía de enmascaramiento y filtrado[16] utilizado en imágenes donde el principal objetivo es inutilizar o eliminar la información adicional anexionada a la imagen como puede ser la titularidad de los derechos de autor o la información del usuario que ha adquirido los derechos de uso de la misma para así poder hacer un uso fraudulento de la misma.

Ataque pasivo

En un ataque pasivo, la finalidad no es solo la detección del uso de la esteganografía, sino el intento de descifrado de la información esteganografiada en el estego-objeto original sin la modificación de este. Pero para considerar roto un sistema esteganográfico y considerarse un estegoanálisis pasivo como positivo solo se ha de detectar la existencia de un mensaje oculto y no el descifrado del propio mensaje.

Existen dos tipos principales de estegoanálisis pasivo:

- **Estegoanálisis pasivo manual.**

Esta técnica se basa en una comparación de forma manual entre el portador y el estego-objeto en busca de diferencias que indiquen la existencia de datos ocultos. Para poder llevar a cabo esta comparación es necesario disponer de ambos ficheros, algo no muy común, y aun habiendo detectado cambios entre ambos ficheros, esto solo indicaría la existencia de datos ocultos, pero no recuperaría la información esteganografiada.

- **Estegoanálisis pasivo estadístico.**

Esta técnica se basa en la utilización de software especializado que sin necesidad de tener el fichero portador, es capaz de buscar en el estego-objeto las huellas que dejan los programas más utilizados de esteganografiado para así detectar la existencia de información oculta.

[16] Explicado en el punto 2.2. TÉCNICAS DIGITALES DE ESTEGANOGRAFIADO

3. ESTEGANOGRAFÍA EN CÓDIGO C

En este apartado vamos a explicar los distintos algoritmos de esteganografiado que se han diseñado para solventar los distintos problemas que se fueron encontrando durante la investigación así como los métodos que utilizan para su funcionamiento.

- **Algoritmo de esteganografiado en ficheros de código C:** Este es el algoritmo pensado inicialmente y utiliza como estego-objeto un código escrito en lenguaje de programación "C". A partir de la realización y estudio de este algoritmo fueron surgiendo dos inconvenientes que se solucionaron en los otros dos algoritmos. Primero, el problema de la persistencia de la información cuando únicamente se dispone del fichero ejecutable (y no del código fuente). Y segundo, la escasa capacidad de almacenamiento de información esteganografiada.

- **Algoritmo de esteganografiado en ficheros de ejecutables .EXE:** Este algoritmo se diseñó para solucionar el problema de la persistencia de la información esteganografiada. El método de esteganografiado es distinto al de esteganografiado en ficheros de código C y una vez completado el estego-objeto, este se compila para obtener así un ejecutable EXE el cual no puede ser modificado, pero del que si es posible obtener la información esteganografiada.

- **Algoritmo de esteganografiado ampliado en ficheros de código .C:** Este último algoritmo se diseñó para solucionar el problema de la capacidad de esteganografiado que tenían los códigos escritos en lenguaje de programación "C" y se basa en utilizar simultáneamente los dos algoritmos descritos anteriormente para utilizar así los puntos de esteganografiado de los dos algoritmos aumentando de este modo la capacidad de esteganografiado.

3.1. ALGORITMO DE ESTEGANOGRAFIADO EN FICHEROS DE CÓDIGO .C

Para realizar el proceso de esteganografiado el algoritmo realiza una serie de procedimientos con el código base en C, que se utilizara como portador, para poder operar con él. Dichos procedimientos consisten en convertir el código base en C de entrada en lo que a partir de este momento llamaremos Código *Cero*, dicho código está formado por una copia del código base de entrada al cual se le han establecido a valor cero o valor inicial todos sus *puntos de esteganografiado*.

Para general el *Código Cero* se recorrerá el fichero base de entrada copiando en el nuevo fichero todas las líneas leídas que no contengan ningún *punto de esteganografiado*. Dichos *punto de esteganografiado* son las operaciones matemáticas, las expresiones en sentencias condicionales "IF" y los bucles "WHILE" y "DO...WHILE". Cuando se encuentra alguna línea que contiene uno de estos puntos, se establece su valor a cero y después se copia la línea al nuevo fichero, para establecer dicho valor lo que se hace es esteganografiar un cero en dicho punto. (Los métodos de esteganografiado se explicaran más adelante)

Finalizada la creación del *Código Cero* para usarlo de *portador* lo único que resta por hacer es calcular la capacidad de dicho código, lo que nos muestra cuantos bits de información son capaces de esteganografiarse en dicho código.

Una vez obtenido el *Código Cero* y la capacidad ya se puede esteganografiar cualquier mensaje que en código binario no supere dicha capacidad. Para ello el algoritmo solicitará al usuario el mensaje que desea esteganografiar y lo transformara a código binario, almacenando esteganograficamente esa serie de unos y ceros en los distintos *puntos de esteganografiado* del *Código Cero,* la forma de esteganografiar cada bits se explica en el apartado siguiente. El código resultante de dicha modificación sobre el *Código Cero* no es más que el código base con el mensaje ya esteganografiado o lo que es lo mismo, el *Código Esteganografiado.*

Para el **desesteganografiado** de un código, se ha de obtener su *Código Cero* al igual que al esteganografiar, la diferencia que para el desesteganografiado lo que sucede a continuación no es una modificación sobre dicho código sino que lo que el procedimiento lleva a cabo es una comparación entre el *Código Esteganografiado* y el *Código Cero.* Así se puede comprobar las variaciones en los *puntos de esteganografiado* y obtener la secuencia correcta de desesteganografiado de los mismos, debido a que el esteganografiado de varias operaciones anidadas puede variar el orden de los *puntos de esteganografiado* que la forman. Una vez ordenado el *Código Esteganografiado*, se procede al desesteganografiado de sus *puntos de esteganografiado,* dando como resultado una cadena de bits la cual forma el mensaje.

Explicación del método del cálculo de la capacidad

El cálculo de la capacidad se consigue con la fórmula:
Capacidad = Nº expresiones *2 + Nº Símbolos matemáticos.

En las expresiones de las sentencias condicionales y de los bucles, mediante el uso de negaciones, se pueden almacenar 2 bits de información y en las operaciones matemáticas tan solo 1 bits, por lo tanto la suma de todas las operaciones matemáticas y del doble de las expresiones nos da como resultado la capacidad de esteganografiado del fichero, que es el número total de bits que se pueden esteganografiar en dicho fichero.

Explicación del método de esteganografiado para ficheros .C

- **Esteganografiado de operaciones matemáticas:**

Se localizan en el código las operaciones matemáticas buscando en él los símbolos **+**, **-** y *****.

```
1.  int main() {
2.  Int A=1, B=2, c=0, d=3, e=0, f=0;
3.  Boolean g, h;
4.  printf("La suma es: %d",(A + B));
5.  If (c = d)
6.  printf("La suma es correcta");
7.  e=c;
8.  If (e>f){
9.  while(g && h){
10. g=leer_estado_uno();
11. h=leer_estado_dos();}
12. }
13. return 0;
14. }
```

Operación matemática encontrada: (A+B)

En cada operación matemática encontrada en el código, se pueden almacenar 1 bit expresado como la comparación de sus operandos.

Si el primer operando es **alfabéticamente mayor** al segundo, el valor esteganográfico de la operación es 1, sino, su valor es 0. Para poder mantener el resultado de la operación se ha de modificar el símbolo matemático de acuerdo al nuevo orden de los operandos.

(A+B) → A < B esteganografiado = 0
(B+A) → B > A esteganografiado = 1

Para el caso de operaciones matemáticas anidadas, se mantiene el orden de las operaciones originales, esteganografiándose primero lo contenido dentro de los paréntesis desde el interior hacia el exterior.

(A + B) + C → Primera operación (A + B) → A < B, valor esteganografiado = 0.
Segunda operación (A +B) + C → (A+B) < C, valor esteganografiado = 0.
Valor esteganografiado total de la operación = 00.

(B + A) + C → Primera operación (B + A) → B > A, valor esteganografiado = 1.
Segunda operación (B + A) + C → (B + A) < C, valor esteganografiado = 0.
Valor esteganografiado total de la operación = 01.

Ejemplo

Esteganografiado

Operación Inicial: (C=A-B)

Strcmp (A, B) = -1. (A < B) El operando **A** es menor al operando **B**, entonces, el valor esteganográfico es 0, por lo tanto la operación está en ***Código Cero***.

Operación Código Cero: (C=A-B)

Si se quiere almacenar un 1, lo único que hay que hacer es cambiar el orden de los operandos.

Operación Esteganografiada inicial: (C=B-A)

Strcmp (B, A) = 1. (B >A) El operando **B** es mayor al operando **A**, entonces, el valor esteganográfico de **B-A** es 1.

Para mantener el mismo resultado en la operación esteganografiada que en la inicial, al haber cambiado el orden de los operandos hay que cambiar el signo de la operación.

Valor Operación Inicial: (C=A-B) **≠** Valor Operación Esteganografiada inicial (C=B-A)

Al ser la operación una resta y haber cambiado el orden de los operandos:

1º Cambiamos el símbolo de la operación.
(C=B-A) → (C=B+A)
2º Cambiamos el signo del operando cambiado de orden.
(C=B+A) → (C=-B+A)

Valor Operación Inicial: (C=A-B) **=** Valor Operación Esteganografiada final (C=-B+A)

Operación Esteganografiada final (C=-B+A)

Desesteganografiado

Operación Inicial: (C=-B + A)
Strcmp (B, A) = 1. (B > A) El operando **B** es mayor al operando **A**, entonces, el valor esteganográfico es 1.

- **Esteganografiado de expresiones en sentencias condicionales "IF" y bucles "WHILE" y "DO...WHILE"**

Las expresiones de las sentencias condicionales y de los bucles están compuestas de operaciones comparativas o booleanas, las cuales pueden ser modificadas manteniendo su significado y su correcta funcionalidad.

Se busca en el código la palabra "**IF**" o "**WHILE**" y se analiza de qué tipo es la operación:

Operaciones de comparación:

En cada bucle se pueden almacenar 2 bits, uno que depende de si el bucle contiene el símbolo de negación "!" y otro que depende del orden de los operandos.

Ejemplo:

IF (A > B) → esteganografiado = **00**, no tiene negación y el primer operando es menor al segundo.

IF ! (A < B) → esteganografiado = **10**, tiene negación y el primer operando es menor al segundo.

IF (B < A) → esteganografiado = **01**, no tiene negación y el primer operando es mayor al segundo.

IF ! (B < A) → esteganografiado = **11**, tiene negación y el primer operando es mayor al segundo.

Los bucles de comparación que contienen operaciones matemáticas anidadas del tipo **IF ((A+B) <C)**, primero se analiza y esteganografía la **operación matemática** y luego el **bucle de comparación**.

Ejemplo:

IF ((A+B) < C)→ esteganografiado = **000**, el primer operando de la suma "**A**" es menor al segundo operando "**B**"; la comparación con "**C**" no tiene negación y el operando compuesto **(A+B)** es menor al tercer operando "**C**".

IF ((B+A) < C)→ esteganografiado = **001**, el primer operando de la suma "**B**" es mayor al segundo operando "**A**"; la comparación con "**C**" no tiene negación y el operando compuesto **(B+A)** es menor al tercer operando "**C**".

IF (C > (A+B))→ esteganografiado = **010**, el primer operando de la suma "**A**" es menor al segundo operando "**B**"; la comparación con "**C**" no tiene negación y el operando compuesto **(A+B)** es menor al tercer operando "**C**".

IF (C > (B+A))→ esteganografiado = **011**, el primer operando de la suma "**B**" es mayor al segundo operando "**A**"; la comparación con "**C**" no tiene negación y el operando compuesto **(B+A)** es menor al tercer operando "**C**".

IF !((A+B) > C)→ esteganografiado = **100**, el primer operando de la suma "**A**" es menor al segundo operando "**B**"; la comparación con "**C**" tiene negación y el operando compuesto **(A+B)** es menor al tercer operando "**C**".

IF !((B+A) > C)→ esteganografiado = **101**, el primer operando de la suma "**B**" es mayor al segundo operando "**A**"; la comparación con "**C**" tiene negación y el operando compuesto **(B+A)** es menor al tercer operando "**C**".

IF !(C < (A+B))→ esteganografiado = **110**, el primer operando de la suma "**A**" es menor al segundo operando "**B**"; la comparación con "**C**" tiene negación y el operando compuesto **(A+B)** es menor al tercer operando "**C**".

IF !(C < (B+A))→ esteganografiado = **111**, el primer operando de la suma "**B**" es mayor al segundo operando "**A**"; la comparación con "**C**" tiene negación y el operando compuesto **(B+A)** es menor al tercer operando "**C**".

Operaciones relacionales:

En cada bucle relacional "igual a" (==) y "no igual a" (!=), se puede almacenar 1 bit que depende del orden de los operandos.

Si el primer operando es **alfabéticamente mayor** al segundo, el valor esteganográfico de la operación es 1, sino vale 0.

Ejemplo:

IF (A == B) → esteganografiado = **0**, el primer operando es menor al segundo.
IF (B == A) → esteganografiado = **1**, el primer operando es mayor al segundo.

IF (A != B) → esteganografiado = **0**, el primer operando es menor al segundo.
IF (B != A) → esteganografiado = **1**, el primer operando es mayor al segundo.

Operaciones lógicas:

En cada bucle lógico && (AND) y || (OR), se puede almacenar 1 bit que depende del orden de los operandos.

Si el primer operando es **alfabéticamente mayor** al segundo, el valor esteganográfico de la operación es 1, sino vale 0.

Ejemplo:

IF (A && B) → esteganografiado = **0**, el primer operando es menor al segundo.
IF (B && A) → esteganografiado = **1**, el primer operando es mayor al segundo.

IF (A || B) → esteganografiado = **0**, el primer operando es menor al segundo.
IF (B || A) → esteganografiado = **1**, el primer operando es mayor al segundo.

3.2. ALGORITMO DE ESTEGANOGRAFIADO EN FICHEROS DE EJECUTABLES .EXE

Tras realizar el análisis del compilador y de múltiples pruebas se concluyó que los cambios que se realizaban en el código durante el esteganografiado en código C no eran perdurables, debido al procedimiento de optimización que realiza el compilador para generar el ejecutable EXE. La única solución que se ha encontrado a este problema de perdurabilidad es la de utilizar como medio de esteganografiado únicamente las cabeceras de las funciones y de los procedimientos escritos en el código C a esteganografiar, puesto que si el código se compila en modo *Debug* (gcc –g) es posible recuperar dicha información del ejecutable.

Antes de realizar el proceso de esteganografiado el algoritmo realiza un procedimiento con el código base en C, que se utilizara como portador, para poder operar con él de una forma más cómoda y eficiente a la hora de desesteganografiar. Dicho procedimiento consisten en ordenar los archivos de cabecera **.h* del código para que la sentencia *"#include<stdio.h>"* sea la última de la lista. Esto sirve para que una vez compilado el código se pueda usar esta sentencia como *token* de inicio de cabeceras, puesto que entre esta última sentencia y el proceso "*main"* del programa se encuentran el resto de procedimientos y funciones del programa esteganografiado. Ahorrándonos así el análisis de partes del código incluidas en el ejecutable por el resto de librerías o por el compilador que no son necesarias para el proceso de desesteganografiado.

Una vez realizado el procedimiento anterior se calcula la capacidad del código, para conocer cuántos bits se pueden almacenar en dicho código, el método para el cálculo de la capacidad de explicar más adelante.

Una vez obtenida la capacidad ya se puede esteganografiar cualquier mensaje, que en código binario, no supere dicha capacidad. Para ello el algoritmo solicitara al usuario el mensaje que desea esteganografiar y lo transformara a código binario, almacenando esteganograficamente esa serie de unos y ceros en los distintos *Pares de Esteganografiado* del *Código Portador*, la forma de esteganografiar cada bits se explica en el apartado siguiente. El código resultante de dicha modificación sobre el *Código Portador* es el *Código Esteganografiado*.

Una vez obtenido el *Código Esteganografiado*, el algoritmo lo compila en modo *debug* mediante una llamada al compilador GCC para obtener así el ejecutable EXE Esteganografiado.

Para el **desesteganografiado** de un ejecutable, se ha de obtener la secuencia de cabeceras de las funciones que forman el programa esteganografiado. Dichas cabeceras gracias al procedimiento de ordenación de sentencia *"#include<xxxx.h>"* se puede obtener de la lectura en modo binario/texto del ejecutable, puesto que estas se encuentran entre los *token* "/include/sys/types.h" y ".main:F", que pertenecen a la declaración del ultimo archivo de cabecera de *stdio.h* y a la declaración del procedimiento *main* respectivamente.

Tras obtener las cabeceras, se procede al desesteganografiado de sus distintos *Pares de Esteganografiado*, dando como resultado la cadena de bits que forman el mensaje esteganografiado inicial.

Explicación del método del cálculo de la capacidad

El cálculo de la capacidad se consigue con la fórmula:

Capacidad = Nº funciones o procedimientos /2 + ∑ (Nº Argumentos de cada función o procedimiento /2).

Para cada función o procedimiento, se podrá almacenar un bit por cada par de argumentos que tenga, puesto que se usa la posición alterable de estos como método esteganográfico.

Al igual que se pueden alterar las posiciones de los argumentos también se pueden alterar las posiciones de las funciones y de los procedimientos pudiendo almacenar un bit por cada par de funciones que haya en el código, excluyendo la función *main.*

En resumen, la capacidad es la suma del número de pares de funciones o procedimientos más la suma del número de pares de argumentos de cada función o procedimiento.

Ejemplo:

Para un programa con el siguiente esqueleto:	La capacidad seria de:
Funcion1 (Arg 1, Arg2, Arg3, Arg4);	4 funciones / 2 = 2 Bits
Funcion2 (Arg 1, Arg2, Arg3);	Funcion1 = 4 argumentos / 2 = 2 Bits
Funcion3 (Arg 1, Arg2);	Funcion2 = 3 argumentos / 2 = 1 Bits
Funcion4 (Arg 1, Arg2, Arg3, Arg4);	Funcion3 = 2 argumentos / 2 = 1 Bits
Main (Arg 1, Arg2, Arg3, Arg4);	Funcion4 = 4 argumentos / 2 = 2 Bits
	Total: 2 + (2 + 1 + 1 + 2) = 8 Bits

Explicación del método de esteganografiado para ejecutables .EXE

- **Esteganografiado de funciones o procedimientos**

Se buscan en el código las cabeceras de las funciones y de los procedimientos, sin incluir la función *main*, y se agrupan de dos en dos, en lo que llamaremos ***Par de Esteganografiado***. Se pueden almacenar 1 bit esteganografiado en cada *Par de Esteganografiado*, expresado como la comparación alfabética de los operandos que la forman.

Una vez agrupadas las funciones y procedimientos se analiza cada uno de los *Pares de Esteganografiado* para comprobar si existieran llamadas a funciones o procedimientos que impidieran su alteración en el orden de aparición o declaración en el código. Debido a la posibilidad de que dentro de una función (Función padre) pueda existir la llamada a una o más funciones declaradas previamente (Funciones Hijo), el orden de estas puede no ser alterable, debido a que no puede estar declarada la función padre antes que la función hijo.

Finalizado el análisis y esteganografiado de todos los *Pares de Esteganografiado,* solo se han de copiar en el código, en su nuevo orden, todas las funciones y procedimientos que los forman para obtener el *Código Esteganografiado*.

El esteganografiado de un *Par de Esteganografiado* depende de su análisis, dando este uno de los siguientes resultados:

A. No existe dependencia de llamadas entre los elementos del *Par de Esteganografiado*:

Par 1 (Función 1, Función 2) = Función 2 --\--> Función 1

Al no existir dependencia este *Par de Esteganografiado* se puede esteganografiar sin problemas de la siguiente manera: Si el nombre del primer elemento es **alfabéticamente mayor** al segundo, el valor esteganográfico del *Par de Esteganografiado* es 1, sino vale 0. Para cambiar el valor del *Par de Esteganografiado* solo se ha de invertir el orden de los elementos.

Par 1 (Función 1, Función 2) → Nombre Función 1 > Nombre Función 2 → esteganografiado = 1
Par 1 (Función 2, Función 1) → Nombre Función 2 < Nombre Función 1 → esteganografiado = 0

B. Existe dependencia de llamadas entre los elementos del *Par de Esteganografiado* (La función 2 llama en su código a la función 1):

Par 1 (Función 1, Función 2) = Función 2 -----> Función 1

Se intercambian los elementos del *Par de Esteganografiado* actual con los del siguiente *Par de Esteganografiado* (Función 3 y Función 4) y se analiza la dependencia:

Par 1 (Función 1, Función 2)
Par 2 (Función 3, Función 4)

a) No existe dependencia de llamadas entre los elementos del Par 1 y los del Par 2:

Par 1 (Función 1, Función 2) = Función 2 -----> Función 1
Par 2 (Función 3, Función 4) = Función 4 --\--> Función 3
Función 3 --\--> Función 1
Función 3 --\--> Función 2
Función 4 --\--> Función 1
Función 4 --\--> Función 2

Nuevo Par 1 (Función 1, Función 3) = Función 3 --\--> Función 1
Nuevo Par 2 (Función 2, Función 4) = Función 4 --\--> Función 2

El esteganografiado de estos nuevos pares sin dependencia es idéntico al del apartado **A**

b) Existe dependencia débil de llamadas entre los elementos del Par 1 y los del Par 2:

Par 1 (Función 1, Función 2) = Función 2 -----> Función 1
Par 2 (Función 3, Función 4) = Función 3 -----> Función 1

Nuevo Par 1 (Función 1, Función 4) = Función 4 --\--> Función 1
Nuevo Par 2 (Función 2, Función 3) = Función 3 --\--> Función 2

El esteganografiado de estos nuevos pares sin dependencia es idéntico al del apartado **A**

c) Existe dependencia fuerte de llamadas entre los elementos del Par 1 y los del Par 2:

Par 1 (Función 1, Función 2) = Función 2 -----> Función 1
Par 2 (Función 3, Función 4) = Función 3 -----> Función 2

Si existe este tipo de dependencia, los elementos no pueden variar su posición en el código con lo cual **no es posible la esteganografía** basada en procedimientos y funciones para este código en particular.

- **Esteganografiado de parámetros en funciones o procedimientos**

Se buscan en el código las cabeceras de las funciones y de los procedimientos, sin incluir la función *main*, y se agrupan sus parámetros de dos en dos, en lo que llamaremos ***Par de Parámetros***. Se pueden almacenar 1 bit esteganografiado en cada *Par de Parámetros*, expresado como la comparación alfabética de los operandos que lo forman.

Al no existir ningún tipo de restricción en el orden aparición de los parámetros en las cabeceras de las funciones o procedimientos, este puede ser alterado siempre que se mantenga el mismo orden en las llamadas a dicha función o procedimiento en el resto del código. En resumen, los parámetros de la cabecera de una función o procedimiento y las llamadas a dicha función o procedimiento han de tener el mismo orden.

Cada *Par de Parámetros* se puede esteganografiar sin problemas de la siguiente manera: Si el nombre del primer parámetro es **alfabéticamente mayor** al segundo, el valor esteganográfico del *Par de Parámetros* es 1, sino vale 0. Para cambiar el valor del *Par de Parámetros* solo se ha de invertir el orden de los elementos.

Finalizado el esteganografiado de todos los *Pares de Parámetros,* solo se han de copiar en el código las nuevas llamadas y las nuevas cabeceras de las funciones y de los procedimientos con los parámetros reordenados para obtener el *Código Esteganografiado*.

Ejemplo:

Función inicial: **Función (Parametro_1, Parametro_2, Parametro_3)**

El nombre del parámetro 1 es **alfabéticamente mayor** al nombre del parámetro 2 el valor esteganográfico es 0:

Par 1 (Parametro_1, Parametro_2) → Parametro_1 < Parametro_2 → esteganografiado = 0

Para almacenar un 1 como valor esteganográfico se cambia el orden de los parámetros:

Par 2 (Parametro_2, Parametro_1) → Parametro_2 > Parametro_1 → esteganografiado = 1

Función esteganografiada: **Función (Parametro_2, Parametro_1, Parametro_3)**

3.3. ALGORITMO DE ESTEGANOGRAFIADO AMPLIADO EN FICHEROS DE CÓDIGO .C

Este último algoritmo de esteganografiado se desarrolló para dotar de una mayor capacidad de esteganografiado a un código. Para ello, el algoritmo realiza los dos métodos de esteganografiado explicados anteriormente sobre el mismo código base en C, tanto el método de esteganografiado normal para código C como el esteganografiado para ficheros EXE.

Al realizarse los dos métodos simultáneamente, aunque se obtiene una mayor capacidad de esteganografiado, se sacrifica la Invisibilidad perceptiva del método, debido a que el método de esteganografiado sobre EXE altera el código C resultante de una manera más evidente frente a una comparación directa entre dos códigos. El tema sobre la invisibilidad de los métodos y su resistencia al estegoanálisis se tratara en los siguientes apartados de esta memoria.

3.4. OPCIONES DE ESTEGANOGRAFIADO

Uno de los principales inconvenientes encontrados en el uso de estos algoritmos es baja la capacidad encontrada en los códigos originales usados como estego-objeto. Debido a esto y a que los códigos originales no pueden ser aumentados de capacidad, se ha optado por la reducción del mensaje a esteganografiar, con el fin de aumentar así la capacidad de esteganografiado de los códigos. Para ello se han definido dos modos de esteganografiado.

- **Opción de esteganografiado de caracteres**

Esta opción es el esteganografiado normal y admite como parte del mensaje a esteganografiar cualquier tipo de carácter ASCII. Usando este método, cada uno de los caracteres del mensaje es pasado a binario necesitando así 8 bits para poder esteganografiar cada carácter del mensaje.

- **Opción de esteganografiado de mayúsculas**

Esta opción es la desarrollada para aumentar la capacidad, pero como todo, tiene una serie de ventajas y de inconvenientes. Este método solo permite que el mensaje a esteganografiar esté formado por letras mayúsculas y esto es debido a que cada una de esas letras mayúsculas que forman el mensaje pasara por un método para reducir su longitud de los 8 bits originales a solo 5 aumentando así la capacidad neta del código original a casi el doble.

Para llevar a cabo esta reducción, partimos de los 8 bits que forman cada carácter en binario y les eliminamos la parte común “010”que tienen todos ellos en su parte izquierda. La cadena resultante es el array que se esteganografiará en el código como mensaje esteganografiado. Al desesteganografiar, se le volverán a añadir la cabecera “010” a cada uno de los caracteres para así volver a recuperar el mensaje original.

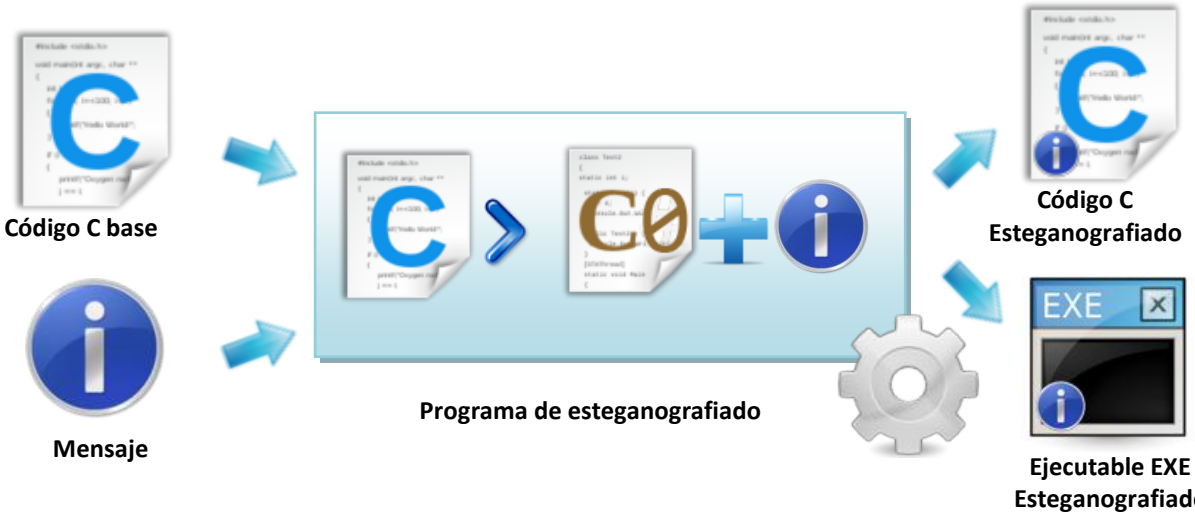
Mensaje original	IVAN R	Longitud
Array original	01001001 01010110 01000001 01001110 00100000 01010010	48 bits
Array reducido	01001 10110 00001 01110 00000 10010	30 bits

Tabla 5: Ejemplo reducción tamaño del mensaje

4. APLICACIONES PRÁCTICAS DEL ALGORITMO DE ESTEGANOGRAFÍA

Con el cambio de *portador* se van a poder obtener una serie de nuevos usos para la esteganografía que no solo se centran en la ocultación de información, sino que usan a este mismo principio para una serie de aplicaciones prácticas como son las siguientes:

ESTEGANOGRAFIADO DE INFORMACIÓN:
Se tiene la capacidad de ocultar información, al igual que pasa con el resto de medios usados para la esteganografía, dentro del propio código C o en el ejecutable EXE resultante de la compilación de dicho código.

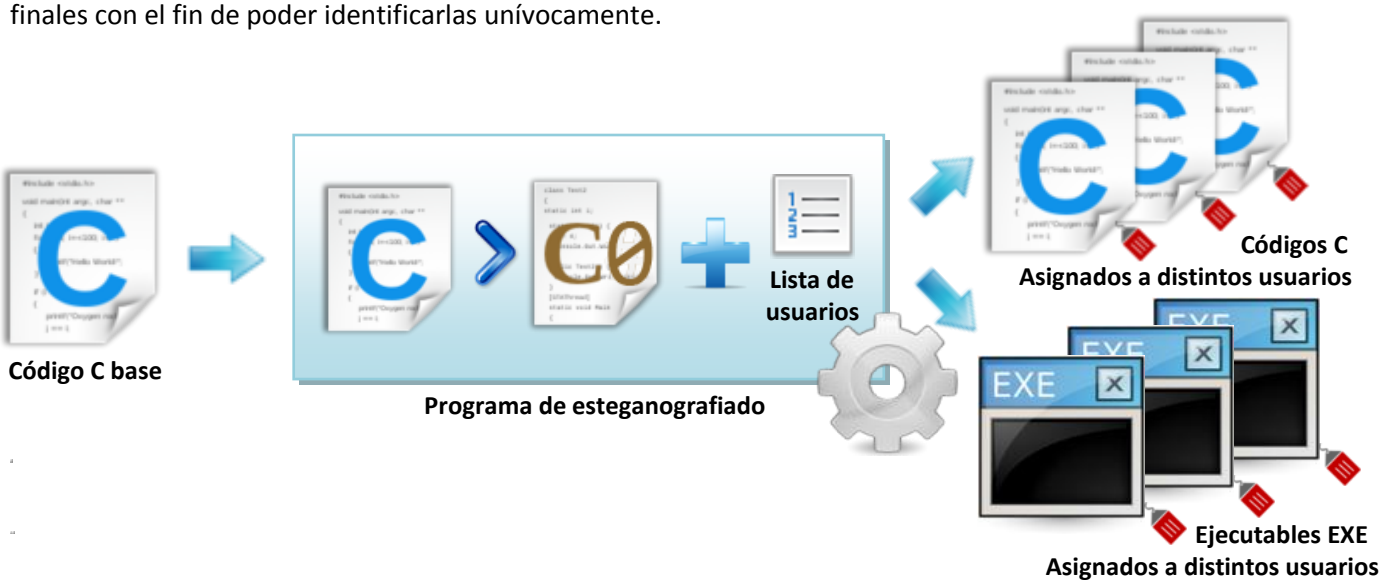


Aplicación: Ocultación de mensajes o características del propio código que no se desean hacer públicas.

FIRMA DIGITAL DE UN CÓDIGO:
Se obtiene la capacidad de firmar el código de manera que esta queda oculta al usuario final.

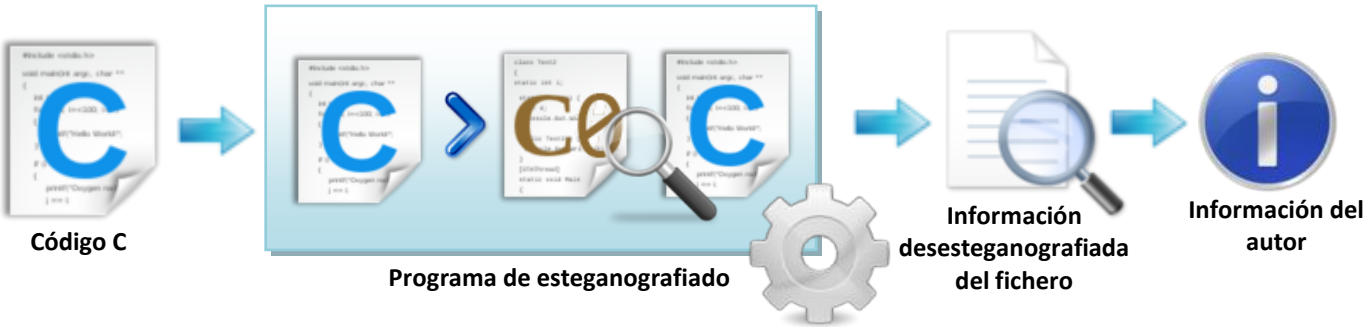
Aplicación: Debido a que tanto el autor como el resto de las propiedades de la mayoría de los archivos informáticos y por ende de los archivos de código C, puede ser fácilmente alterable, ocultar la firma, de manera que esta no pueda ser eliminada o modificada de un código, nos da la posibilidad de demostrar la legitimidad de un documento si esto fuera necesario.

IDENTIFICACIÓN DE DISTINTAS COPIAS DE UN MISMO CÓDIGO:
Se obtiene la capacidad de asignar cada una de las copias de un mismo código a distintos usuarios finales con el fin de poder identificarlas unívocamente.



Aplicación: La ocultación de los datos del usuario final del código en el propio código de manera que cada una de las copias pueda ser relacionada unívocamente con cada uno de los usuarios nos permite tomar acciones contra posibles usos no deseados del código, como por ejemplo su difusión o copia ilegal.

IDENTIFICACIÓN DE UN CÓDIGO DENTRO DE OTRO:
Se obtiene la capacidad de firmar el código de manera oculta y de poder ser este identificado cuando forma a su vez, parte de un código mayor.



Aplicación: La firma oculta, de manera que esta no pueda ser eliminada o modificada, de un código, nos da la posibilidad de demostrar la legitimidad de un fragmento de un código que forma a su vez parte de un código mayor si esto fuera necesario. Como por ejemplo el uso dentro de un programa con ánimo de lucro y sin permiso de un fragmento de código con licencia de libre distribución.

5. DISEÑO Y PRUEBAS DEL PROGRAMA DE ESTEGANOGRAFIADO

5.1. DISEÑO DEL PROGRAMA

En este apartado vamos a explicar con detalle la solución llevada a cabo para resolver el problema descrito en los anteriores puntos de este documento.

Como el lenguaje utilizado para el desarrollo del programa es el C, el cual no está orientado a objetos, no es posible la realización de un diagrama de clases UML. Por lo que para explicar más claramente este apartado vamos a utilizar los diagramas de flujo.

Para la representación de los diagramas de flujo, se va a seguir la simbología clásica explicada en la siguiente tabla:

Símbolo	Nombre	Descripción
	Terminal	Inicio o final del diagrama de flujo.
	Entrada y salida de datos.	Entrada o salida de información desde o hacia el ordenador.
	Proceso.	Proceso interno realizado por el ordenador como asignación de un valor en la memoria o la ejecución de operaciones matemáticas.
	Decisión	Indica la realización de una comparación entre valores, siguiendo el programa distinta vía en función del caso.
	Línea de flujo.	Secuencia del flujo de procesos.
	Conector	Indica a través de una referencia (número, letra o texto) dónde debe continuar un diagrama de flujo que se interrumpe.
	Subprograma	El programa principal pasa a ejecutar todas las instrucciones contenidas en el subprograma para una vez terminadas continuar el flujo.

Tabla 6: Leyenda del diagrama de flujo

El siguiente diagrama de flujo representa de una forma esquemática el funcionamiento general del programa y sus distintos modos de funcionamiento. En los siguientes apartados explicaremos con más detalle el funcionamiento de cada modo.

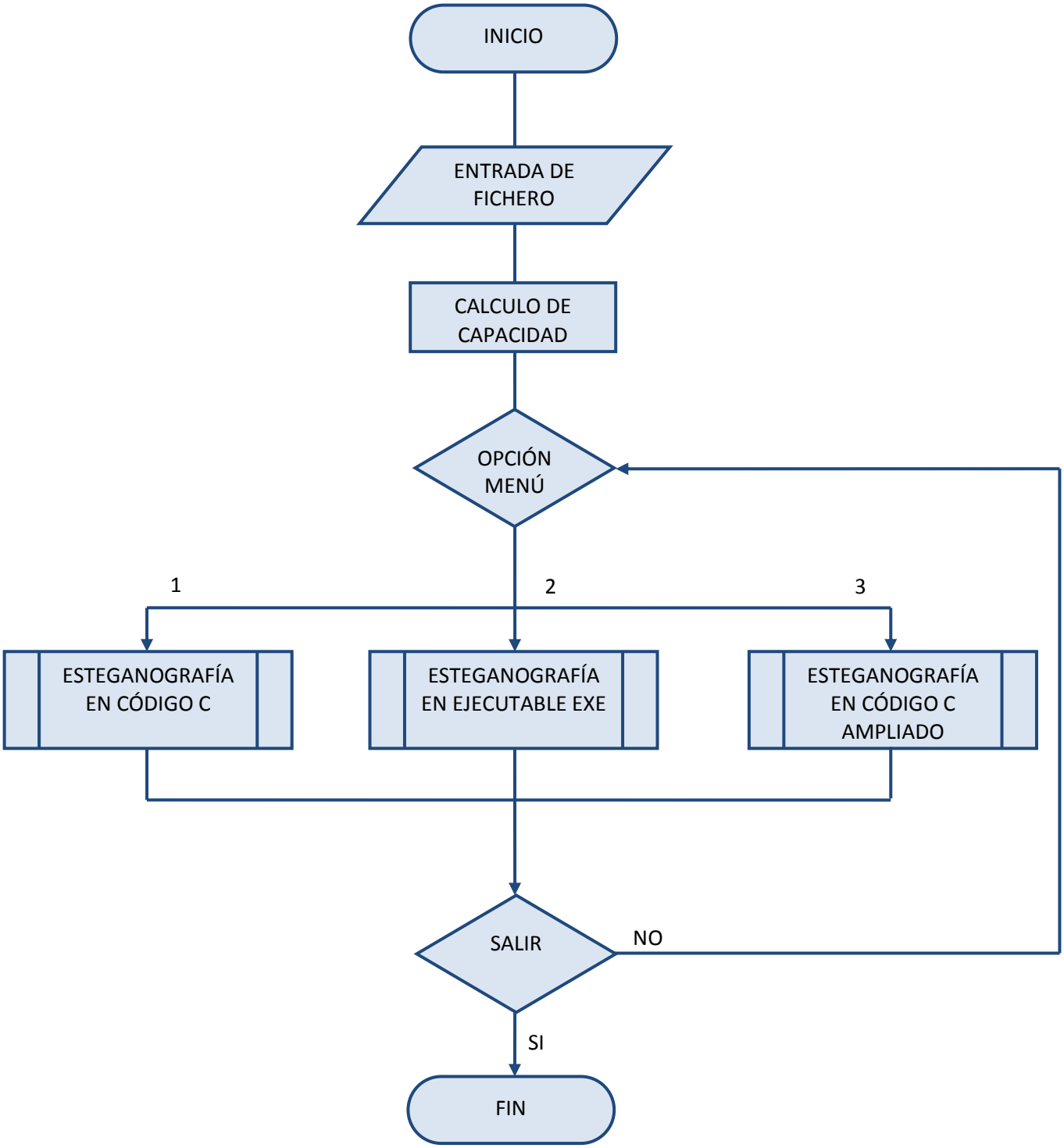


Diagrama de flujo general del programa

ESTEGANOGRAFÍA EN CÓDIGO C

En este apartado vamos a explicar la parte correspondiente al punto 3.1.ALGORITMO DE ESTEGANOGRAFIADO EN FICHEROS DE CÓDIGO .C del documento, el cual se basa principalmente en la manipulación de las operaciones matemáticas, de comparación y relacionales existentes dentro del código sin modificar el resto del fichero.

Para llevar a cabo eso, el programa lee todo el código y lo descompone en un array de "operaciones" enlazadas entre sí cuya estructura es la siguiente:

```
struct str_operacion{
// Datos operacion en lectura
        char nombre[5];
        char variable1[50];
        char valor_var1[50];
        char variable2[50];
        char valor_var2[50];
        char simbolo[3];
        int  negativo;

// Datos operacion ordenados
        char Ord_variable1[50];
        char Ord_valor_var1[50];
        char Ord_variable2[50];
        char Ord_valor_var2[50];
        char Ord_simbolo[3];
        int  Ord_mayor;
        int  Ord_negativo;

// Datos operacion esteganografiados
        char Est_variable1[50];
        char Est_valor_var1[50];
        char Est_variable2[50];
        char Est_valor_var2[50];
        char Est_simbolo[3];
        int  Est_negativo;

        struct str_operacion *op_sig;
};
```

Parte que almacena los datos de la operación en su estado original

Parte que almacena los datos de la operación ordenada

Parte que almacena los datos de la operación esteganografiada

Estructura de operación sacada del código del programa

Esta estructura de "operación" puede estar formada por operaciones sencillas o a su vez puede estar formada por otras operaciones:

```
*******************************
Operacion:

nombre: #A.
-------------------------------
variable1: feof(f).
valor variable1: feof(f).
variable2: 1.
valor variable2: 1.
-------------------------------
variable1 ord: 1.
valor variable1 ord: 1.
variable2 ord: feof(f).
valor variable2 ord: feof(f).
-------------------------------
operacion reducida: (feof(f) != 1).
operacion :         (feof(f) != 1).
operacion ordenada: (1 != feof(f)).
operacion esteganografiada: (  ).

*******************************

nombre: #B.
-------------------------------
variable1: enc.
valor variable1: enc.
variable2: 'n'.
valor variable2: 'n'.
-------------------------------
variable1 ord: 'n'.
valor variable1 ord: 'n'.
variable2 ord: enc.
valor variable2 ord: enc.
-------------------------------
operacion reducida: (enc == 'n').
operacion :         (enc == 'n').
operacion ordenada: ('n' == enc).
operacion esteganografiada: (  ).

*******************************

nombre: #C.
-------------------------------
variable1: #A.
valor variable1: (feof(f)!=1).
variable2: #B.
valor variable2: (enc=='n').
-------------------------------
variable1 ord: #B.
valor variable1 ord: ('n'==enc).
variable2 ord: #A.
valor variable2 ord: (1!=feof(f)).
-------------------------------
operacion reducida: (#A && #B).
operacion :
           ((feof(f)!=1) && (enc=='n')).
operacion ordenada:
           (('n'==enc) && (1!=feof(f))).
operacion esteganografiada: (  ).

*******************************
```

La operación A, es una operación sencilla leída en el código original como:
`(feof(f) != 1)`

Pero cuya ordenación es:
`(1 != feof(f))`

La operación B, también es una operación sencilla, leída en el código original como:
`(enc == 'n')`

Pero cuya ordenación es:
`('n' == enc)`

La operación C, por su parte, es una operación múltiple formada por las operaciones A y B. Leída en el código original como:
`((feof(f)!=1) && (enc=='n'))`

Pero cuya ordenación es:
`(('n'==enc) && (1!=feof(f)))`

Ejemplo de operaciones sacadas del log del programa

Una vez obtenidas todas las operaciones que forman el código, se genera el código cero poniendo todos los valores de esteganografiado de las operaciones a cero.

FICHERO DE CÓDIGO C ORIGINAL				
Operación	A (a + b)	B (c = d)	C (f < e)	D (h && g)
Valor esteganografiado	**0**	**0**	**1**	**1**

Tabla 7: Operaciones del fichero de código original

```
1.  int main() {
2.  Int a=1, b=2, c=0, d=3, e=0, f=0;
3.  Boolean g=TRUE, h=TRUE;
4.  printf("La suma es: %d",(a + b));
5.  If (c = d)
6.  printf("La suma es correcta");
7.  e=c;
8.  If (f<e){
9.  while(h && g){
10. g=leer_estado_uno();
11. h=leer_estado_dos();}
12. }
13. return 0;
14. }
```

Ejemplo de fichero de código C

FICHERO DE CÓDIGO CERO				
Operación	A (a + b)	B (c = d)	C (e > f)	D (g && h)
Valor esteganografiado	**0**	**0**	**0**	**0**

Tabla 8: Operaciones del fichero de código Cero

```
1.  int main() {
2.  Int a=1, b=2, c=0, d=3, e=0, f=0;
3.  Boolean g, h;
4.  printf("La suma es: %d",(a + b));
5.  If (c = d)
6.  printf("La suma es correcta");
7.  e=c;
8.  If (e>f){
9.  while(g && h){
10. g=leer_estado_uno();
11. h=leer_estado_dos();}
12. }
13. return 0;
14. }
```

Código Cero del fichero de ejemplo

Después, el programa puede seguir con uno de los dos modos posibles de funcionamiento, el de esteganografiado y el de desesteganografiado.

Para esteganografiar, se le solicita al usuario la cadena correspondiente al mensaje que se desea esteganografiar, esta cadena puede ser de caracteres o de letras mayúsculas.

De la cadena solicitada, se obtiene un array de números binarios formados tanto por el mensaje a esteganografiar como por la opción de esteganografiado elegida.

	Caracteres	Binario
Cadena a esteganografiar	Ivan	00100100101110110011000010110111
Opción esteganografiado	Cadena de caracteres	0
Array a esteganografiar		000100100101110110011000010110111

Tabla 9: Composición del array de información a esteganografiar

Una vez obtenido el array binario, para esteganografiarlo, se cambia el valor de esteganografiado de todas las operaciones que forman el código cero por los valores del array a esteganografiar.

FICHERO DE CÓDIGO CERO				
Operación	A (a + b)	B (c = d)	C (e > f)	D (g && h)
Valor esteganografiado	**0**	**0**	**0**	**0**
Array a esteganografiar	**0**	**0**	**0**	**1**

Tabla 10: Operaciones del fichero de código Cero

```
1.  int main() {
2.  Int a=1, b=2, c=0, d=3, e=0, f=0;
3.  Boolean g, h;
4.  printf("La suma es: %d",(a + b));
5.  If (c = d)
6.  printf("La suma es correcta");
7.  e=c;
8.  If (e>f){
9.  while(g && h){
10. g=leer_estado_uno();
11. h=leer_estado_dos();}
12. }
13. return 0;
14. }
```

Código Cero del fichero de ejemplo

FICHERO DE CÓDIGO C ESTEGANOGRAFIADO				
Operación	A (a + b)	B (c = d)	C (e > f)	D (h && g)
Valor esteganografiado	**0**	**0**	**0**	**1**

Tabla 11: Operaciones del fichero de código esteganografiado

```
1.  int main() {
2.  Int a=1, b=2, c=0, d=3, e=0, f=0;
3.  Boolean g, h;
4.  printf("La suma es: %d",(a + b));
5.  If (c = d)
6.  printf("La suma es correcta");
7.  e=c;
8.  If (e>f){
9.  while(h && g){
10. g=leer_estado_uno();
11. h=leer_estado_dos();}
12. }
13. return 0;
14. }
```

Código esteganografiado del fichero de ejemplo

Una vez realizado esto, el código resultante es el código original con el mensaje del usuario esteganografiado.

Para desesteganografiar la información, lo que hay que hacer es realizar a la inversa los pasos para esteganografiar.

Primero obtenemos todas las operaciones que forman el código esteganografiado y uniendo todos los valores de esteganografiado de las operaciones obtenemos el array desesteganografiado.

FICHERO DE CÓDIGO C ESTEGANOGRAFIADO				
Operación	A (a + b)	B (c = d)	C (e > f)	D (h && g)
Valor esteganografiado	**0**	**0**	**0**	**1**

Tabla 12: Operaciones del fichero de código esteganografiado

```
1.  int main() {
2.  Int a=1, b=2, c=0, d=3, e=0, f=0;
3.  Boolean g, h;
4.  printf("La suma es: %d",(a + b));
5.  If (c = d)
6.  printf("La suma es correcta");
7.  e=c;
8.  If (e>f){
9.  while(h && g){
10. g=leer_estado_uno();
11. h=leer_estado_dos();}
12. }
13. return 0;
14. }
```

Código esteganografiado del fichero de ejemplo

Array desesteganografiado		0001001001011101100110000010110111
	Caracteres	**Binario**
Opción esteganografiada	Cadena de caracteres	**0**
Cadena esteganografiada	Ivan	**001001001011101100110000010110111**

Tabla 13: Composición del array de información desesteganografiada

El primer bit del array indica cual fue el método de esteganografiado que se utilizó, el este caso al ser un cero, el método fue el de cadena de caracteres, con lo cual no hay que hacer ningún paso más con el array por lo que el resto de los bits forman en mensaje.

El siguiente diagrama de flujo representa de una forma esquemática los distintos pasos que realiza el subprograma de esteganografiado en código C en sus dos modos de funcionamiento.

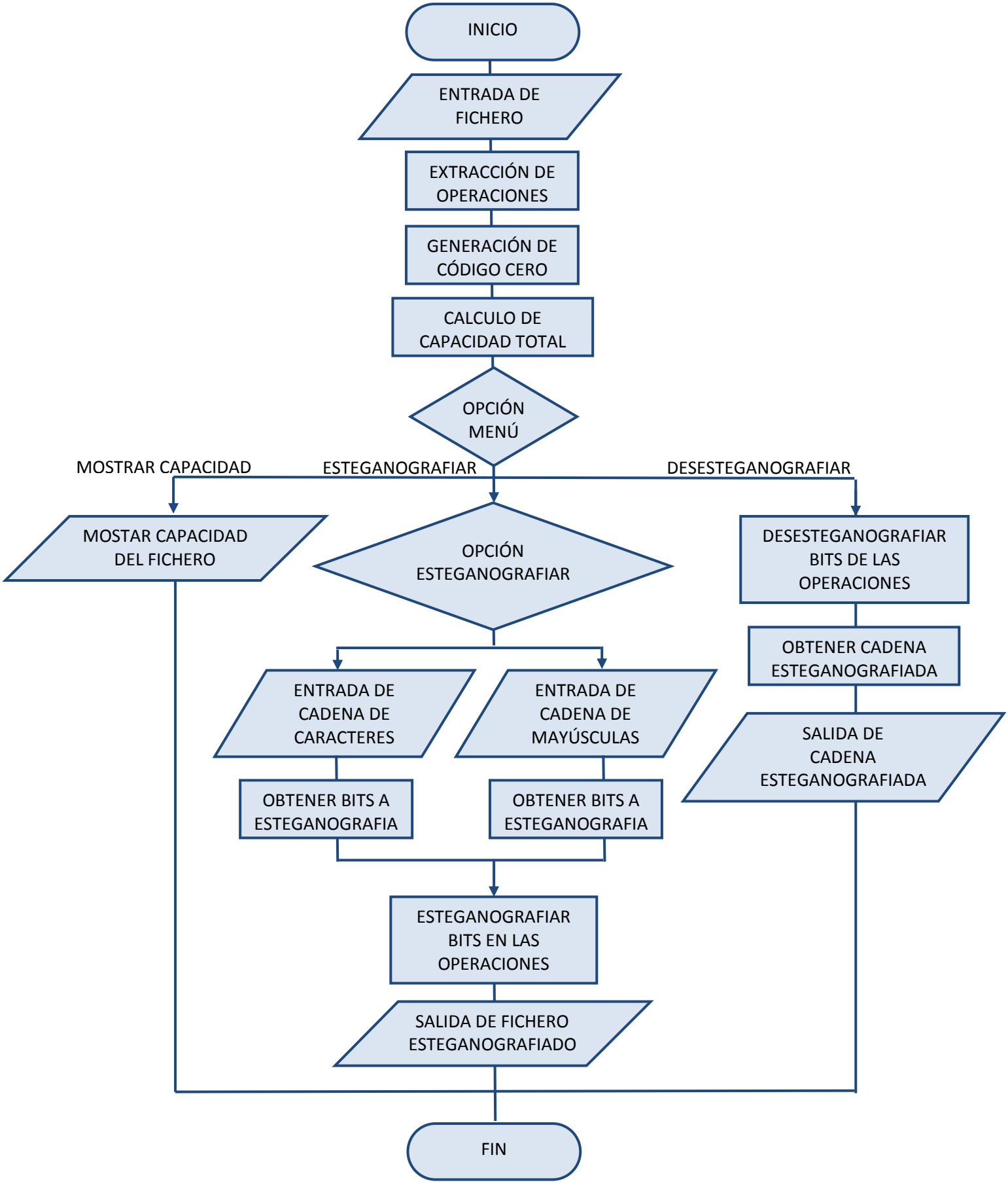
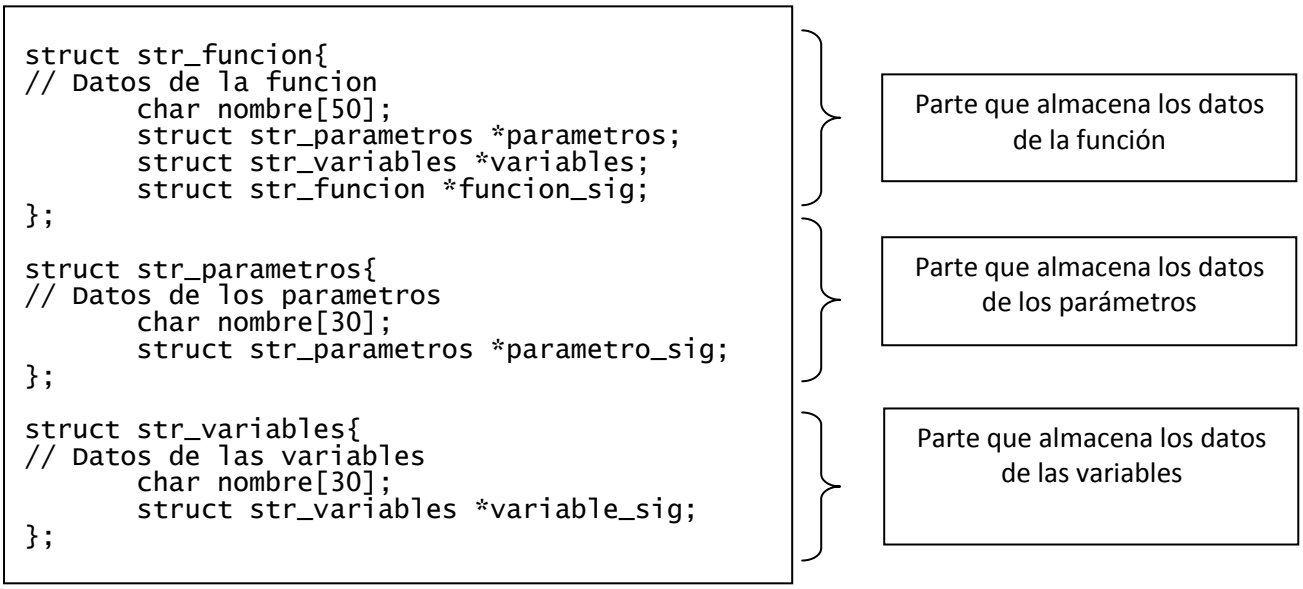


Diagrama de flujo del subprograma de esteganografiado en código C

ESTEGANOGRAFÍA EN EJECUTABLE EXE

En este apartado vamos a explicar la parte correspondiente al punto 3.2.ALGORITMO DE ESTEGANOGRAFIADO EN FICHEROS DE EJECUTABLES EXE del documento, el cual se basa principalmente en la manipulación de las funciones que forman el código.

Para llevar a cabo eso, el programa descompone el código en un array de "funciones" enlazadas entre sí cuya estructura es la siguiente:



Estructura de función sacada del código del programa

A diferencia del método anterior que primero obtenía toda la estructura del fichero y después la utilizaba para esteganografiar o desesteganografiar, este método solo almacena en memoria la estructura mostrada anteriormente con los datos mínimos de las funciones y no todo el código de las propias funciones. Puesto que esto sería poco eficiente en términos de memoria y capacidad.

Para esteganografiar al igual que en el método anterior, se le solicita al usuario la cadena correspondiente al mensaje que se desea esteganografiar, esta cadena puede ser de caracteres o de letras mayúsculas.

De la cadena solicitada, se obtiene un array de números binarios formados tanto por el mensaje a esteganografiar como por la opción de esteganografiado elegida.

	Caracteres	Binario
Cadena a esteganografiar	IVAN	01001001010110010000010100111 0
Opción esteganografiado	Cadena de mayúsculas	1
Array a esteganografiar reducido		1010011011000000101110

Tabla 14: Composición del array de información a esteganografiar

Una vez obtenido el array a esteganografiar, el programa va recorriendo el código C y trabajando por separado con cada par de funciones que encuentra.

FICHERO DE CÓDIGO C ORIGINAL		
	Función 1	**Función 2**
Nombre	media2	Par
Parámetros	(float n2, float n1)	(int numero)
Variables	float resultado	int aux, div

Tabla 15: Par de funciones del fichero de código original

```
1.  float media2(float n2, float n1){
2.   float resultado;
3.   resultado=(n1+n2)/2;
4.   return(resultado);
5.  }
6.  int par(int numero){
7.  Int aux, div;
8.  aux = numero;
9.  div =2;
10.  if((aux%div)==0)
11.  return(1);
12.  else
13.  return(0);
14. }
```

Ejemplo de fichero de código C

Una vez obtenido el par de funciones, se comienza con su esteganografiado:

Primer paso: Esteganografiado de los nombres de las funciones:

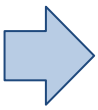
Se comprueba cuál es su valor esteganografiado actual y se compara con el array a esteganografiar, si el valor es el mismo, las funciones se copian con su misma posición en el fichero esteganografiado y si es distinto se copian intercambiando su orden.

FICHERO DE CÓDIGO C ORIGINAL	
Nombres	Media2 < Par
Valor esteganografiado	**0**
Array a esteganografiar	**1**

Tabla 16: Esteganografiado de funciones

```
1.  float media2(float n2, float n1){
2.   float resultado;
3.   resultado=(n1+n2)/2;
4.   return(resultado);
5.  }
6.  int par(int numero){
7.  Int aux, div;
8.  aux = numero;
9.  div =2;
10.  if((aux%div)==0)
11.  return(1);
12.  else
13.  return(0);
14. }
```

Ejemplo de fichero de código C original



```
1.  int par(int numero){
2.  Int aux, div;
3.  aux = numero;
4.  div =2;
5.   if((aux%div)==0)
6.   return(1);
7.   else
8.   return(0);
9.  }
10. float media2(float n2, float n1){
11.  float resultado;
12.  resultado=(n1+n2)/2;
13.  return(resultado);
14. }
```

Fichero de código C esteganografiado parcial

Segundo paso: Esteganografiado de los parámetros de las funciones:

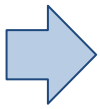
Si el número de parámetros es suficiente, se comprueba cuál es su valor esteganografiado actual y se compara con el array a esteganografiar, si el valor es el mismo, los parámetros de la cabecera se copian con su misma posición en el fichero esteganografiado y si es distinto se copian intercambiando su orden.

FICHERO DE CÓDIGO C ORIGINAL		
	Función 1	**Función 2**
Parámetros	n2 < n1	numero
Valor esteganografiado	**1**	**-**
Array a esteganografiar	**0**	**-**

Tabla 17: Esteganografiado de parametros

```
1.  int par(int numero){
2.  Int aux, div;
3.  aux = numero;
4.  div =2;
5.   if((aux%div)==0)
6.   return(1);
7.   else
8.   return(0);
9.  }
10. float media2(float n2, float n1){
11.  float resultado;
12.  resultado=(n1+n2)/2;
13.  return(resultado);
14. }
```

Fichero de código C esteganografiado parcial



```
1.  int par(int numero){
2.  Int aux, div;
3.  aux = numero;
4.  div =2;
5.   if((aux%div)==0)
6.   return(1);
7.   else
8.   return(0);
9.  }
10. float media2(float n1, float n2){
11.  float resultado;
12.  resultado=(n1+n2)/2;
13.  return(resultado);
14. }
```

Fichero de código C esteganografiado parcial

Cuando se cambia el orden de los parámetros de las funciones, se almacena la nueva cabecera de función para modificar en las llamadas a dicha función en todo el código el orden de los parámetros.

Tercer paso, esteganografiado de las variables de las funciones:

Si el número de variables es suficiente, se comprueba cuál es su valor esteganografiado actual y se compara con el array a esteganografiar, si el valor es el mismo, las variables de la función se copian con su misma posición en el fichero esteganografiado y si es distinto se copian intercambiando su orden.

FICHERO DE CÓDIGO C ORIGINAL		
	Función 1	**Función 2**
variables	resultado	aux < div
Valor esteganografiado	**-**	**0**
Array a esteganografiar	**-**	**1**

Tabla 18: Esteganografiado de variables

```
1.  int par(int numero){
2.  Int aux, div;
3.  aux = numero;
4.  div =2;
5.   if((aux%div)==0)
6.   return(1);
7.   else
8.   return(0);
9.  }
10. float media2(float n1, float n2){
11.  float resultado;
12.  resultado=(n1+n2)/2;
13.  return(resultado);
14. }
```

Fichero de código C esteganografiado parcial

```
1.  int par(int numero){
2.  Int div, aux;
3.  aux = numero;
4.  div =2;
5.   if((aux%div)==0)
6.   return(1);
7.   else
8.   return(0);
9.  }
10. float media2(float n1, float n2){
11.  float resultado;
12.  resultado=(n1+n2)/2;
13.  return(resultado);
14. }
```

Fichero de código C esteganografiado final

Cuarto paso, compilado del código esteganografiado:

Una vez que se ha esteganografiado todo el código C original, el programa realiza una llamada al compilador GCC++ para compilar el código y obtener así el ejecutable EXE compilado.

Para desesteganografiar la información, lo que hay que hacer es realizar a la inversa los pasos para esteganografiar.

El programa obtiene del ejecutable EXE directamente todo el array de "funciones" que forman el código, equivalente al que se utilizó para esteganografiar, con lo que no hay que ir recorriéndolo de poco en poco como cuando se esteganografiaba.

Una vez obtenido el array, vamos comparando los pares de funciones para obtener su código esteganografiado y uniendo todos los valores de esteganografiado de las funciones obtenemos el array desesteganografiado.

FICHERO DE CÓDIGO C ESTEGANOGRAFIADO			
Par de esteganografiado	Funciones (par > media2)	Parámetros (n1 < n2)	Variables (div > aux)
Valor esteganografiado	**1**	**0**	**1**

Tabla 19: Operaciones del fichero de código esteganografiado

1. F –par
2. p-numero
3. v- div
4. v-aux
5. F-media2
6. p-n1
7. p-n2
8. v-resultado

Estructura obtenida del ejecutable EXE
Carpeta: logs/ Estructura_Desteganografiada.txt

El primer bit del array indica cual fue el método de esteganografiado que se utilizó, el este caso al ser un uno, el método fue el de cadena de mayúsculas, con lo cual hay que hacer un paso más con el array para obtener el resto de los bits que forman en mensaje, concatenar delante la cadena "010" a cada grupo de cinco bits.

Array desesteganografiado		1010011011000000101110
	Caracteres	**Binario**
Opción esteganografiada	Cadena de mayúsculas	**1**
Cadena esteganografiada	IVAN	**01001001010101100100000101001110**

Tabla 20: Composición del array de información desesteganografiada

El siguiente diagrama de flujo representa de una forma esquemática los distintos pasos que realiza el subprograma de esteganografiado en ejecutable EXE en sus dos modos de funcionamiento.

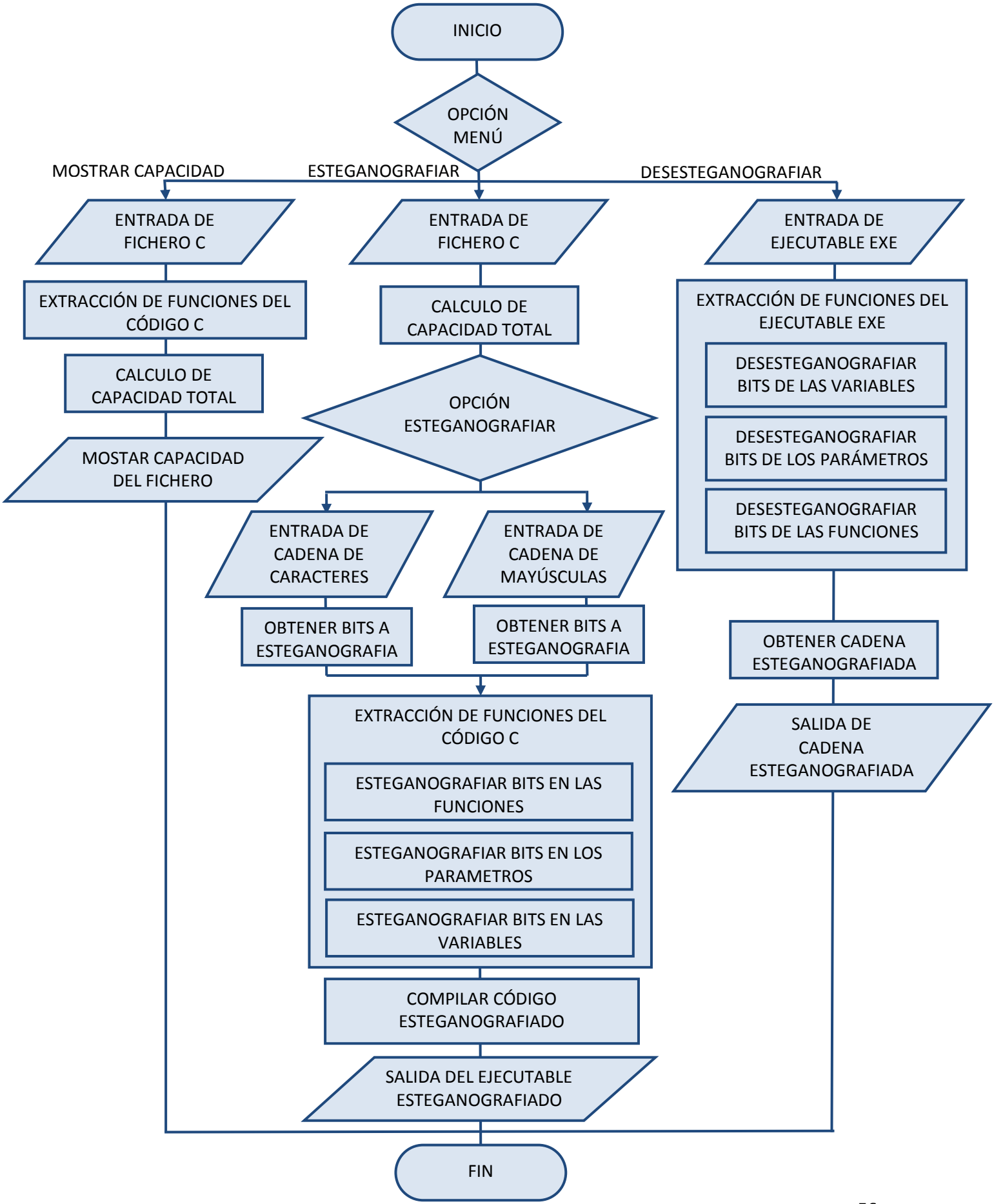
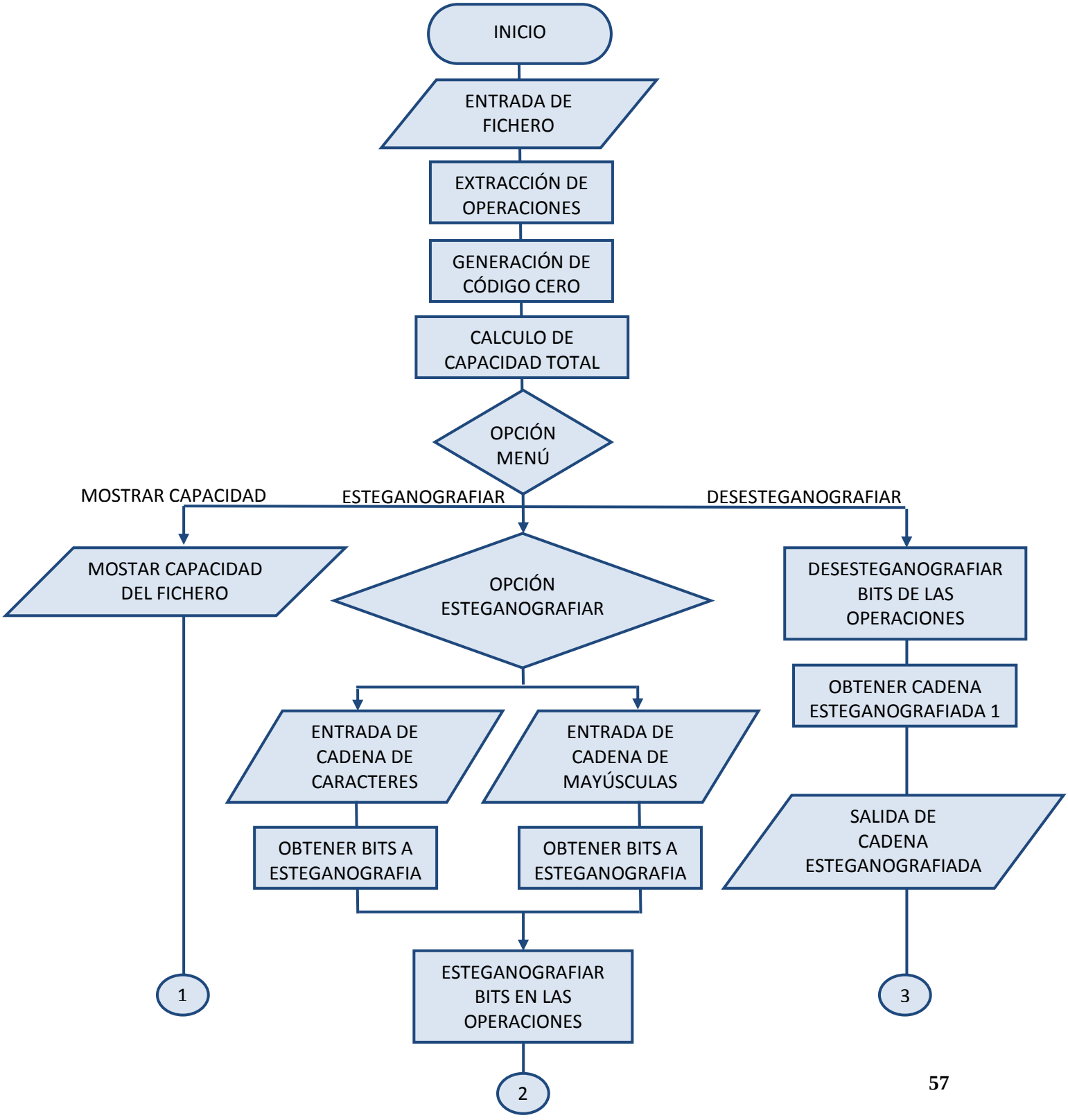


Diagrama de flujo del subprograma de esteganografiado en ejecutable EXE

ESTEGANOGRAFÍA EN CÓDIGO C AMPLIADO

En este apartado vamos a explicar la parte correspondiente al punto 3.3.ALGORITMO DE ESTEGANOGRAFIADO EN FICHEROS DE CÓDIGO .C AMPLIADO del documento. Tal y como en dicho punto se explica, este método se basa en aplicar los dos métodos de esteganografiado explicados anteriormente sobre el mismo código base en C, tanto el método de esteganografiado normal para código C como el esteganografiado para ficheros EXE.

El siguiente diagrama de flujo representa de una forma esquemática los distintos pasos que realiza el subprograma de esteganografiado en código C ampliado en sus dos modos de funcionamiento.



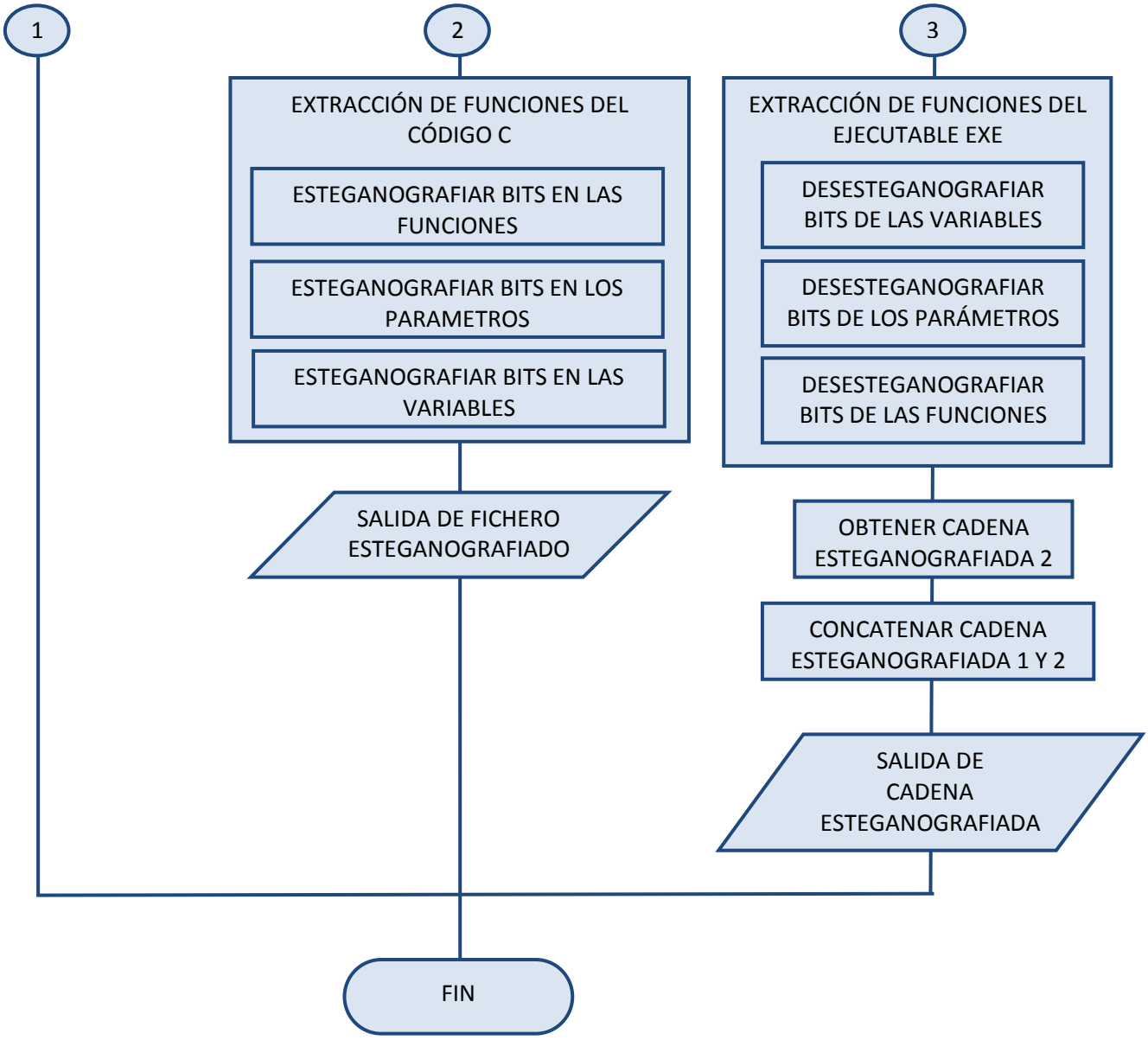


Diagrama de flujo del subprograma de esteganografiado en código C ampliado

5.2. MANUAL Y DIAGRAMA DE NAVEGACIÓN ENTRE MENÚS DEL PROGRAMA

Tras explicar con detalle el funcionamiento del programa vamos a ir explicando un poco las pantallas y menús que lo forman.

El siguiente diagrama está formado por las capturas del programa unidas con flechas que muestran de modo grafico las interacciones del usuario con el propio programa.

PANTALLA DE INICIO:

Pantalla de arranque del programa a modo de bienvenida con el nombre del programa y el del autor.

Ilustración 31: Pantalla de inicio

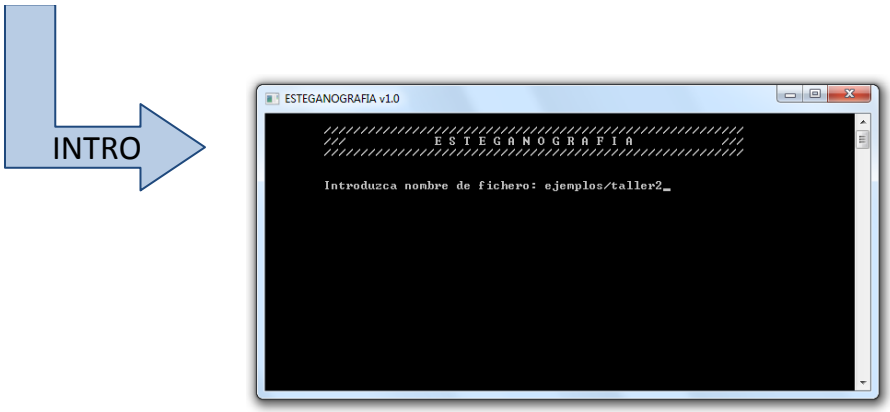


Ilustración 32: Miniatura pantalla de fichero

PANTALLA DE FICHERO:

Pantalla donde se introduce el nombre del fichero que se va a usar para esteganografiar.

Ilustración 33: Pantalla de fichero

PANTALLA DE MENÚ INICIAL:

Pantalla que muestra las distintas opciones para esteganografiar dependiendo de que Estego-objeto queramos obtener.

INTRO

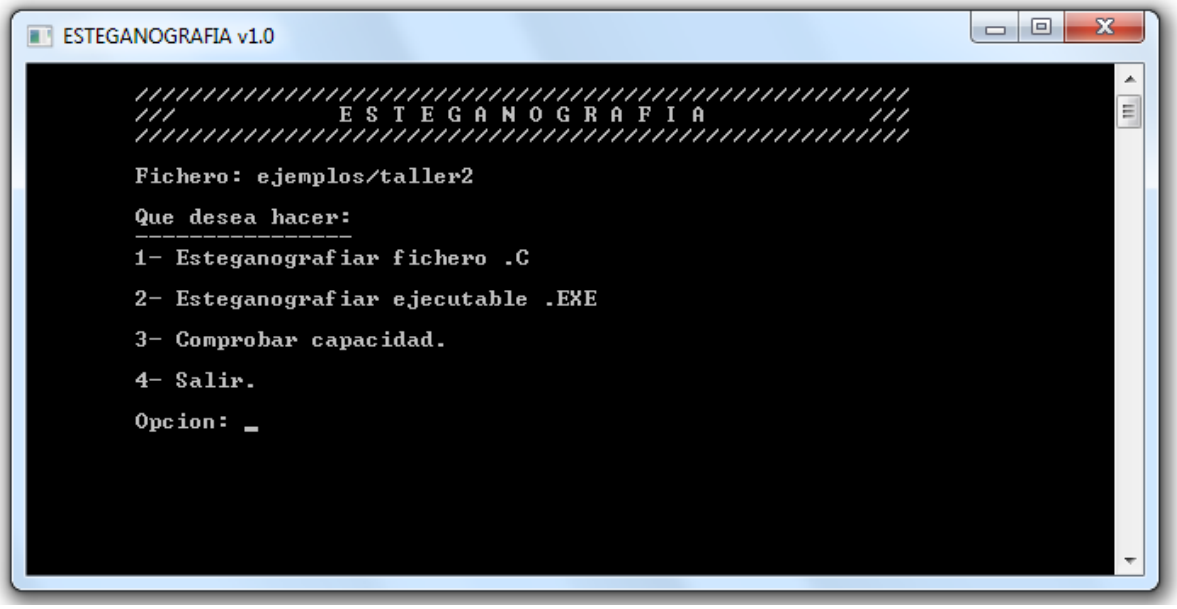


Ilustración 34: Pantalla de menú de esteganografiado

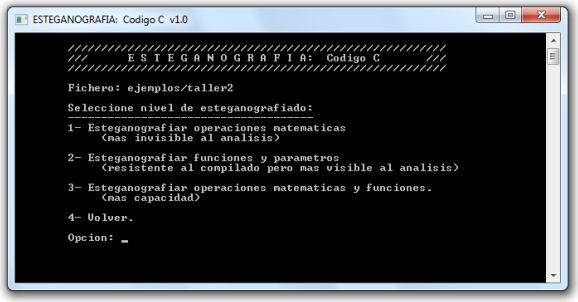


Ilustración 36: Miniatura pantalla de menú de esteganografiado en código C

(Página 62)

1 OPCIÓN 3

2

Ilustración 35: Miniatura pantalla de capacidad total

(Página 61)

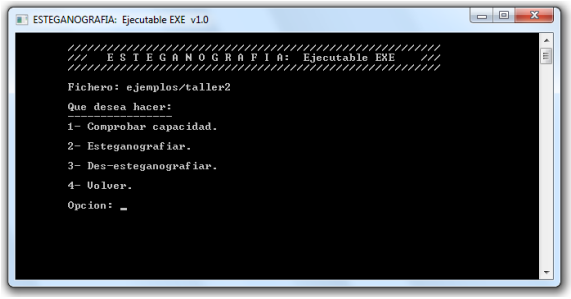


Ilustración 37: Miniatura pantalla de menú de esteganografiado en ejecutable EXE

(Página 67)

PANTALLA DE CAPACIDAD TOTAL:

Pantalla que muestra las capacidades de esteganografiado que tiene el fichero de código C dependiendo de las distintas opciones para esteganografiar.

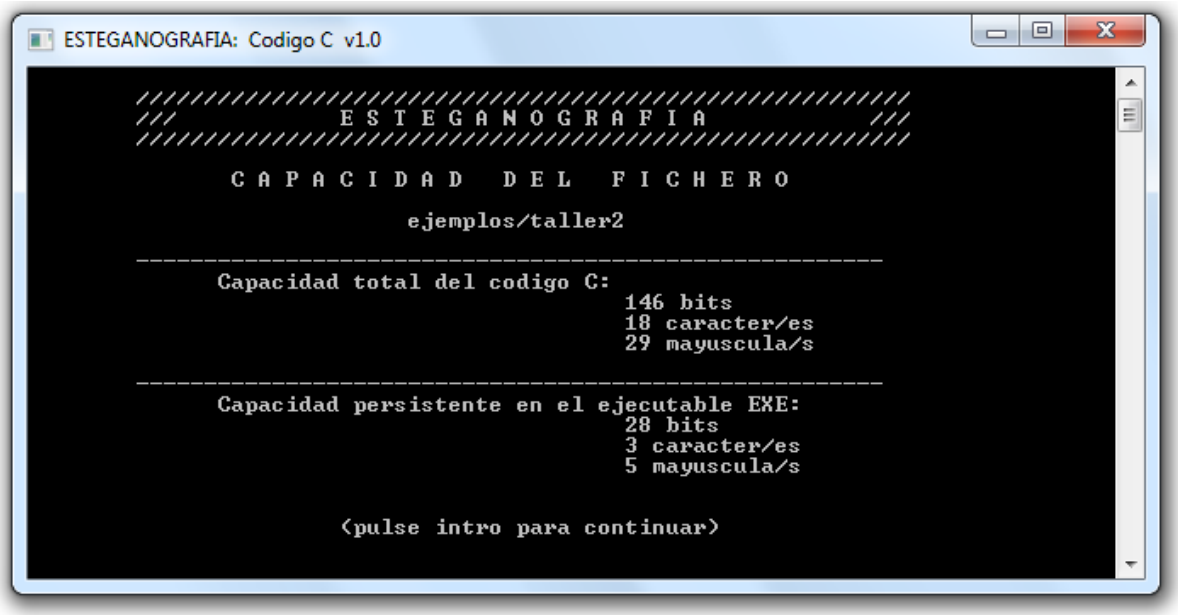


Ilustración 38: Pantalla de menú de esteganografiado en ejecutable EXE

PANTALLA DE MENÚ DE ESTEGANOGRAFIADO EN CÓDIGO C:

Pantalla que muestra las distintas opciones para esteganografiar un fichero de código C que se tienen disponibles. En todas ella, el Estego-objeto que se obtiene es un código C esteganografiado.

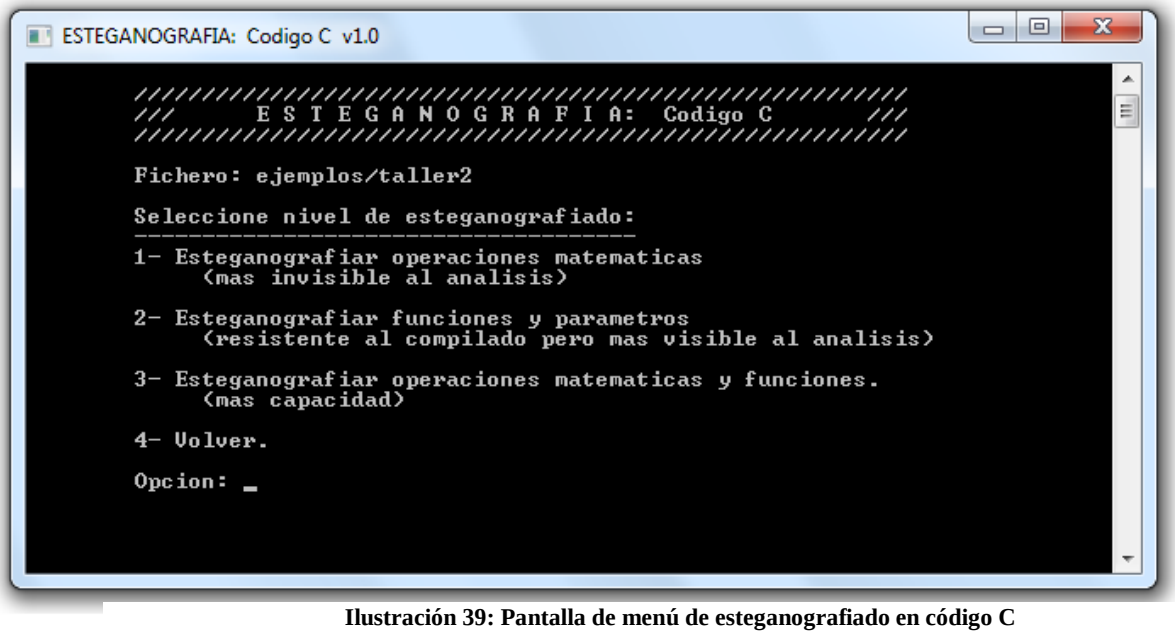


Ilustración 39: Pantalla de menú de esteganografiado en código C

OPCIÓN 1

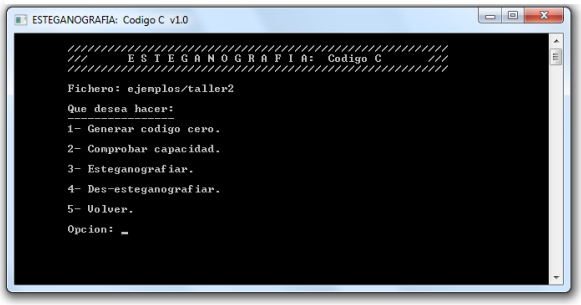


Ilustración 40: Miniatura pantalla de menú de esteganografiado en código C

Modo de esteganografía 1:

Esteganografiado de operaciones matemáticas. Punto 3.1.ALGORITMO DE ESTEGANOGRAFIADO EN FICHEROS DE CÓDIGO .C del documento.

Capacidad: Media
Visibilidad: Baja
Resistencia: Baja

(Página 63)

OPCIÓN 2

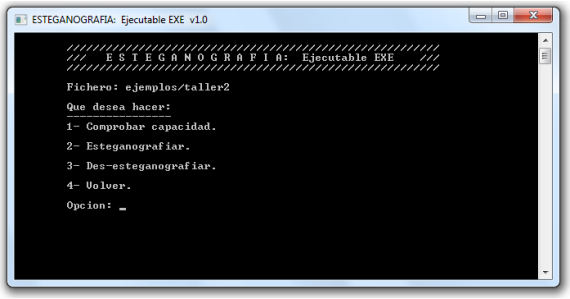


Ilustración 41: Miniatura pantalla de menú de esteganografiado en ejecutable EXE

Modo de esteganografía 2:

Esteganografiado de funciones y parámetros. Punto 3.2.ALGORITMO DE ESTEGANOGRAFIADO EN FICHEROS DE EJECUTABLES .EXE del documento.

Capacidad: Baja
Visibilidad: Alta
Resistencia: Alta

(Página 67)

OPCIÓN 3

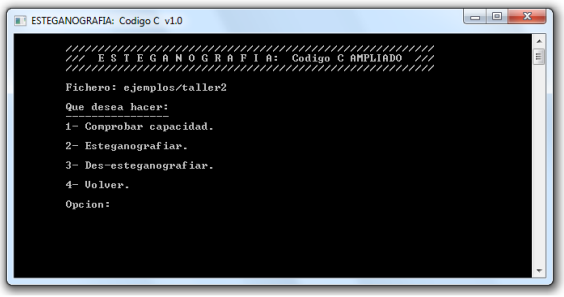


Ilustración 42: Miniatura pantalla de menú de esteganografiado en código C Ampliado

Modo de esteganografía 3:

Esteganografiado combinado de operaciones matemáticas y de funciones y parámetros. Punto 3.3. ALGORITMO DE ESTEGANOGRAFIADO AMPLIADO EN FICHEROS DE CÓDIGO .C del documento.

Capacidad: Alta
Visibilidad: Alta
Resistencia: Baja

(Página 71)

PANTALLA DE MENÚ DE ESTEGANOGRAFIADO EN CÓDIGO C:

Pantalla que muestra las distintas opciones para esteganografiar un fichero de código C usando como Estego-objeto el código C original.

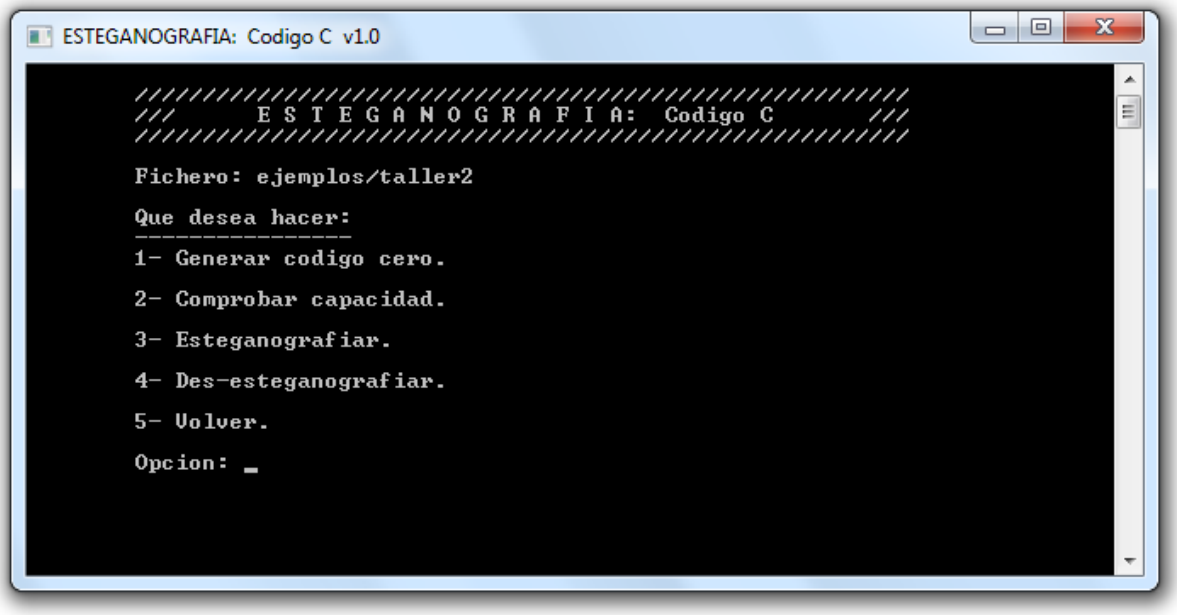


Ilustración 43: Pantalla de menú de esteganografiado en código C

Ilustración 44: Miniatura pantalla de generación de código cero

(Página 64)

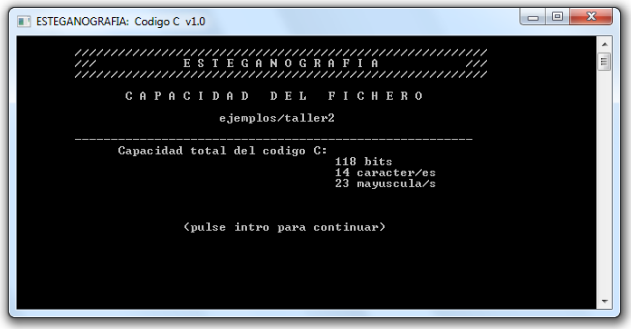
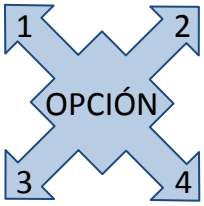


Ilustración 45: Miniatura pantalla de cálculo de capacidad en código C

(Página 64)



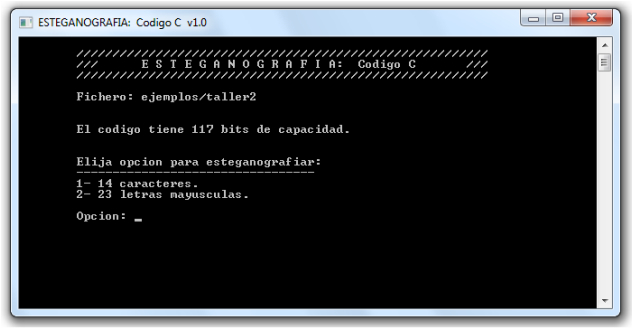


Ilustración 46: Miniatura pantalla de esteganografiado en código C

(Página 65)

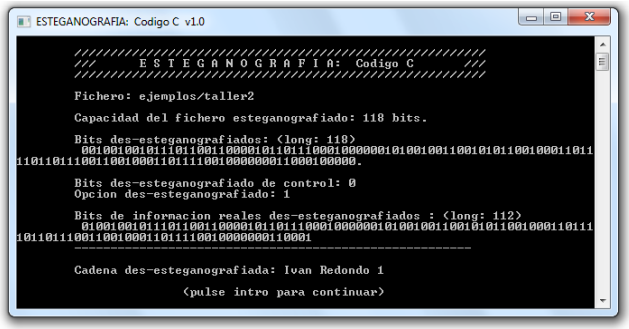


Ilustración 47: Miniatura pantalla de desesteganografiado en código C

(Página 66)

PANTALLA DE GENERACIÓN DE CÓDIGO CERO:

Pantalla que indica que se ha generado correctamente el código cero del fichero que posteriormente se utilizara para el resto de procedimientos.

Ilustración 48: Pantalla de generación de código cero

PANTALLA DE CAPACIDAD TOTAL EN CÓDIGO C:

Pantalla que muestra la capacidad de esteganografiado que tiene el fichero de código C con el modo de esteganografiado de operaciones matemáticas explicado en el punto 3.1.ALGORITMO DE ESTEGANOGRAFIADO EN FICHEROS DE CÓDIGO .C del documento.

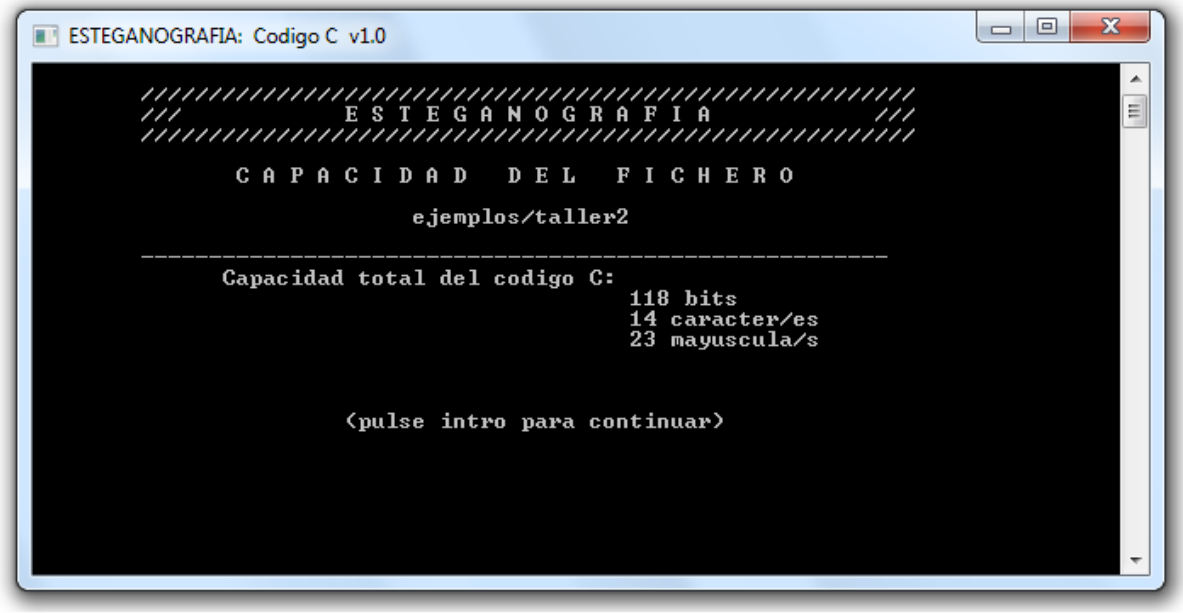


Ilustración 49: Pantalla de cálculo de capacidad en código C

PANTALLA DE ESTEGANOGRAFIADO EN CÓDIGO C:

Pantalla que muestra las dos opciones disponibles para esteganografiar un mensaje en un fichero de código C usando como Estego-objeto el código C original. Las opciones dependen de la capacidad que se desea obtener, el código siempre tiene una capacidad menor para caracteres que para mayúsculas. El programa incluirá en el mensaje a esteganografiar un bit para indicar la opción de esteganografiado elegida.

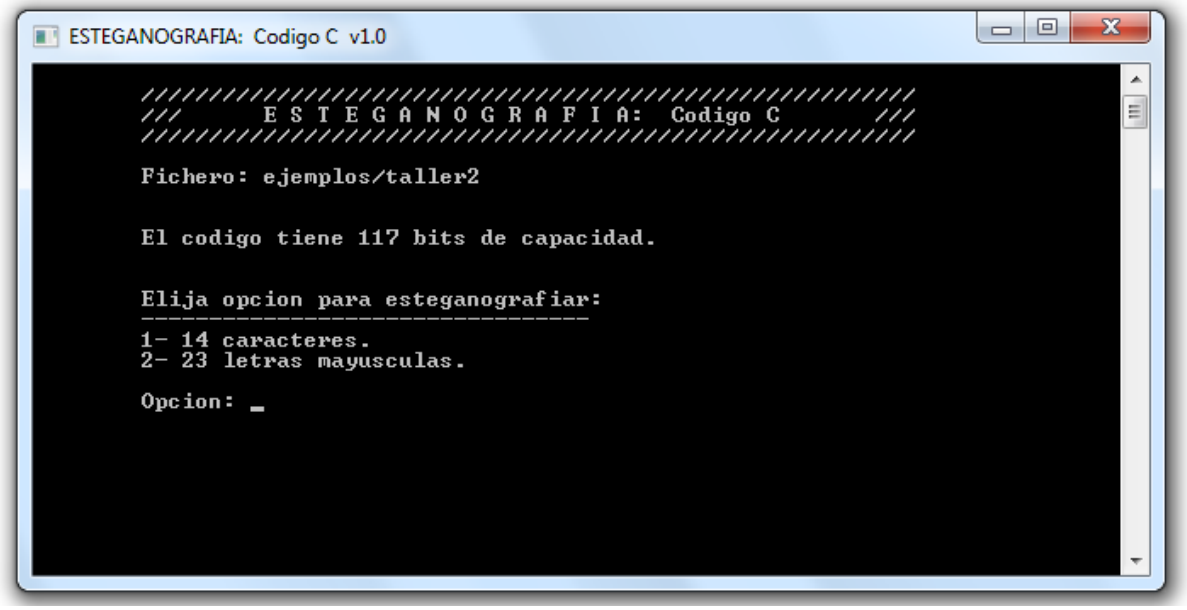


Ilustración 50: Pantalla de esteganografiado en código C

OPCIÓN 1

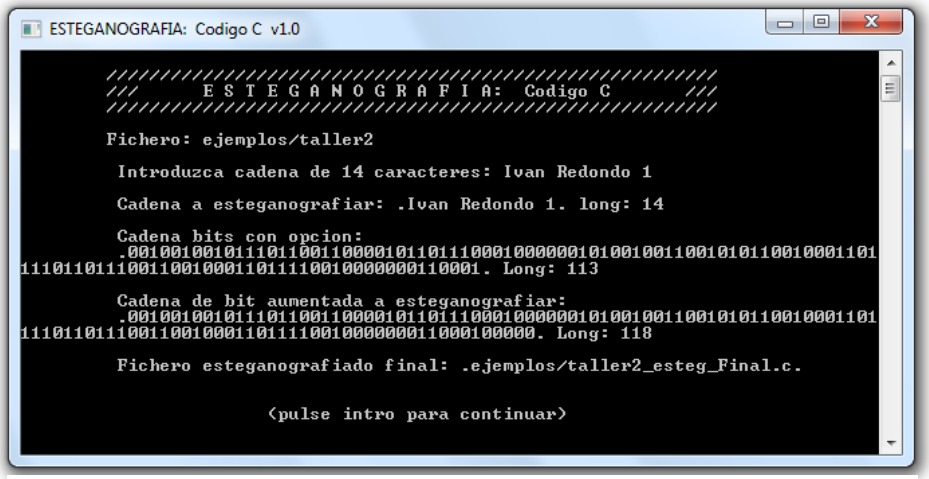


Ilustración 51: Pantalla de esteganografiado de caracteres en código C

Opción 1:
Esteganografiado de caracteres:

Al inicio de la cadena de bits a esteganografiar se incluye un cero para indicar esta opción.

OPCIÓN 2

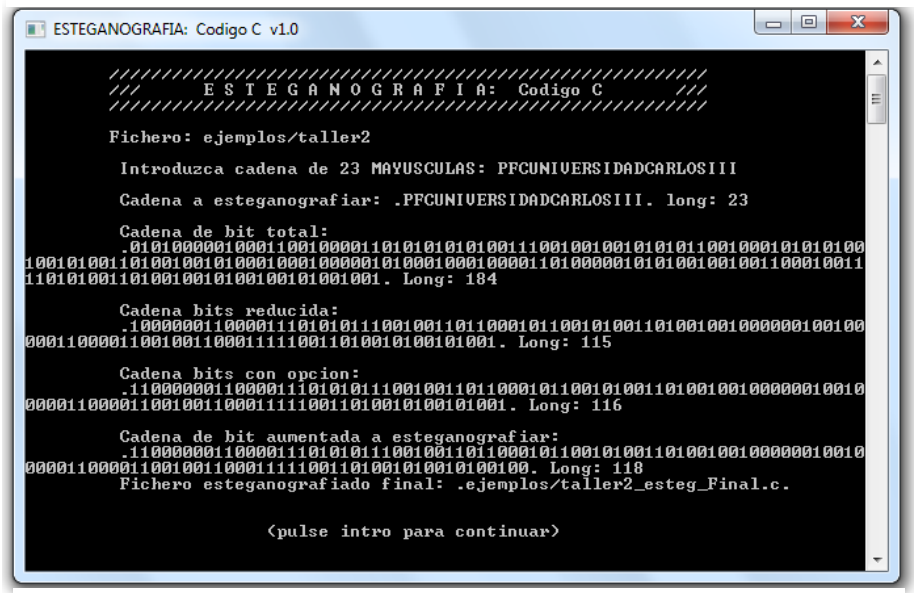


Ilustración 52: Pantalla de esteganografiado de mayúsculas en código C

Opción 2:
Esteganografiado de mayúsculas:

Al inicio de la cadena de bits a esteganografiar se incluye un uno para indicar esta opción.

PANTALLA DE DESESTEGANOGRAFIADO EN CÓDIGO C:

Pantalla que muestra el mensaje desesteganografiado del código C original usado como Estego-objeto. El programa es capaz de reconocer la opción de esteganografiado que se utilizó por lo que no es necesario indicárselo en este paso, para ello utiliza el llamado "bit de control".

En la Ilustración 53 podemos observar como el primer bit desesteganografiado es un cero, lo que le indica al programa que la opción que se utilizó para el esteganografiado fue la de caracteres. Con lo que el programa únicamente tiene que transformar la cadena en binario obtenida a su equivalente en caracteres para obtener el mensaje esteganografiado.

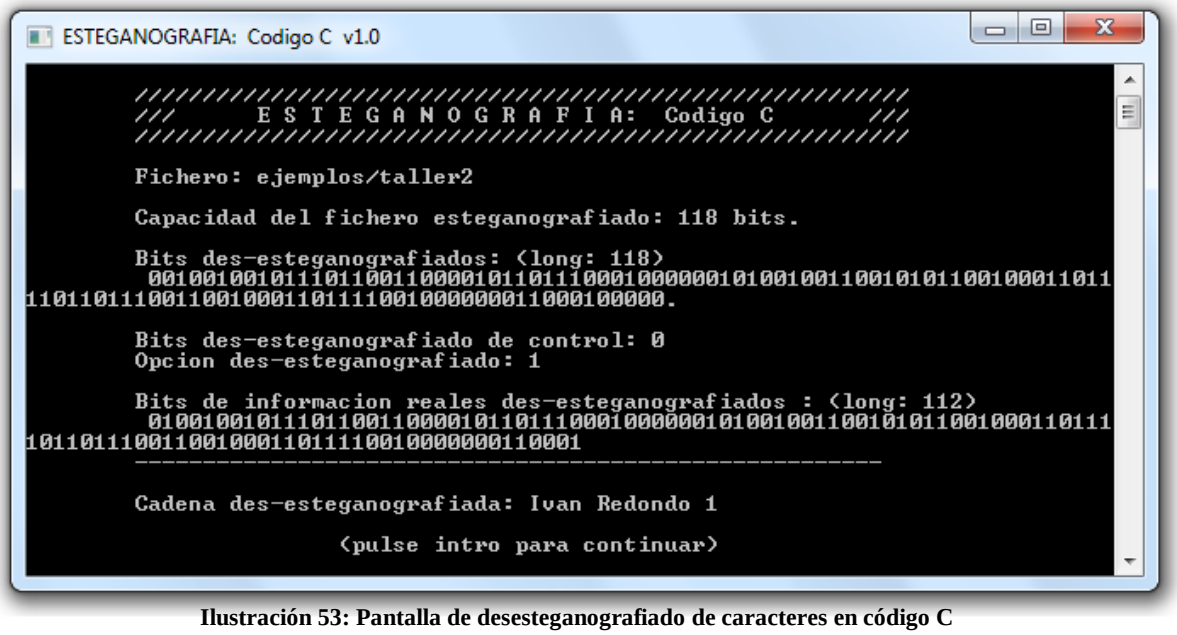


Ilustración 53: Pantalla de desesteganografiado de caracteres en código C

Ahora, en la Ilustración 54 podemos observar el caso contrario, el primer bit desesteganografiado es un uno, lo que le indica al programa que la opción que se utilizó para el esteganografiado fue la de mayúsculas. Con lo que el programa primero tiene que ampliar la cadena reducida en binario obtenida y posteriormente transformarla a una a su equivalente en caracteres para obtener el mensaje esteganografiado.

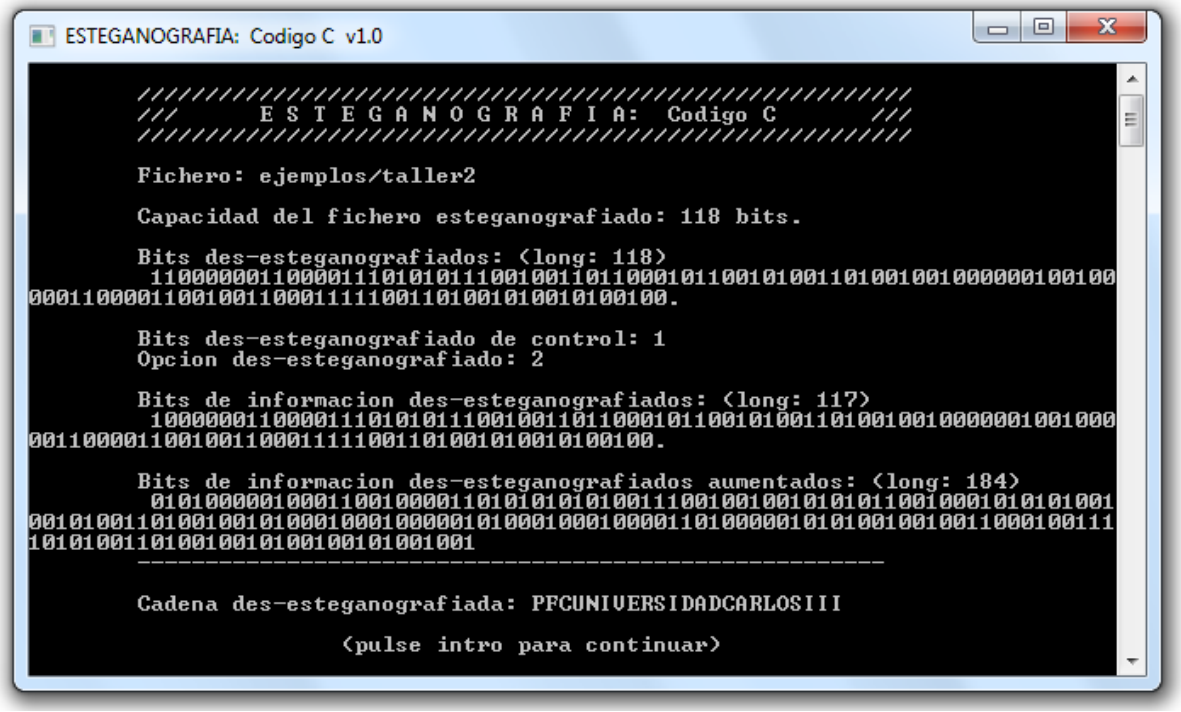


Ilustración 54: Pantalla de desesteganografiado de caracteres en código C

PANTALLA DE MENÚ DE ESTEGANOGRAFIADO EN EJECUTABLE EXE:

Pantalla que muestra las distintas opciones para esteganografiar un fichero de código C usando como Estego-objeto un ejecutable EXE obtenido del código C original.

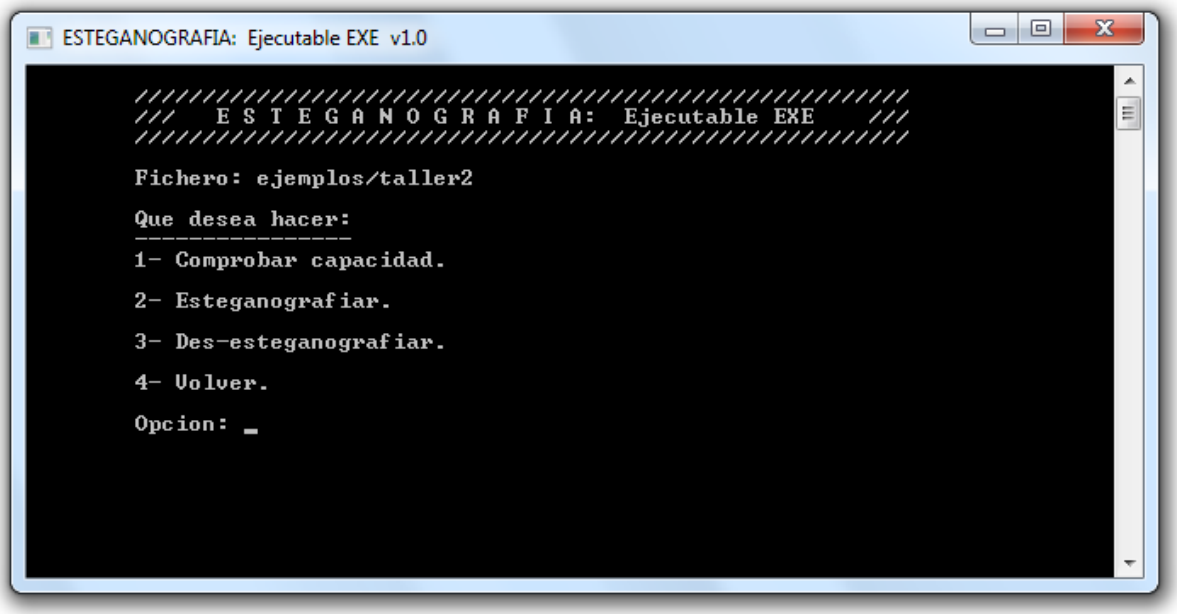


Ilustración 55: Pantalla de menú de esteganografiado en ejecutable EXE

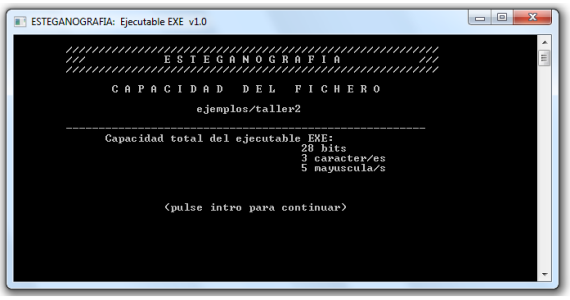
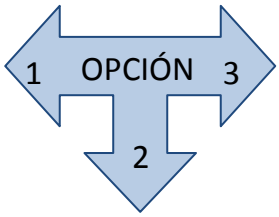


Ilustración 56: Miniatura pantalla de cálculo de capacidad en ejecutable EXE

(Página 68)



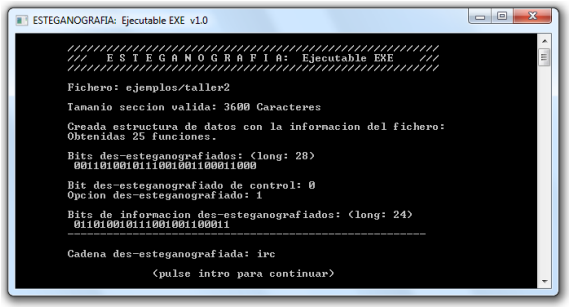


Ilustración 57: Miniatura pantalla de desesteganografiado en ejecutable EXE

(Página 70)

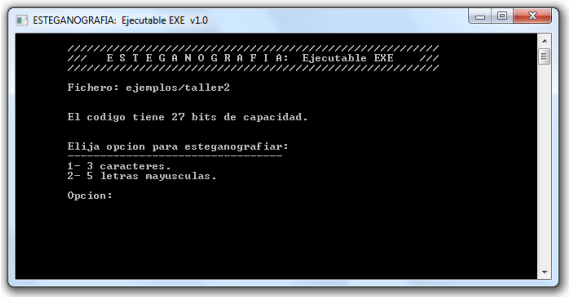


Ilustración 58: Miniatura pantalla de esteganografiado en ejecutable EXE

(Página 68)

PANTALLA DE CAPACIDAD TOTAL EN EJECUTABLE EXE:

Pantalla que muestra la capacidad de esteganografiado que tiene el ejecutable EXE obtenido del código C original con el modo de esteganografiado de funciones y parámetros explicado en el punto 3.2.ALGORITMO DE ESTEGANOGRAFIADO EN FICHEROS DE EJECUTABLES .EXE del documento.

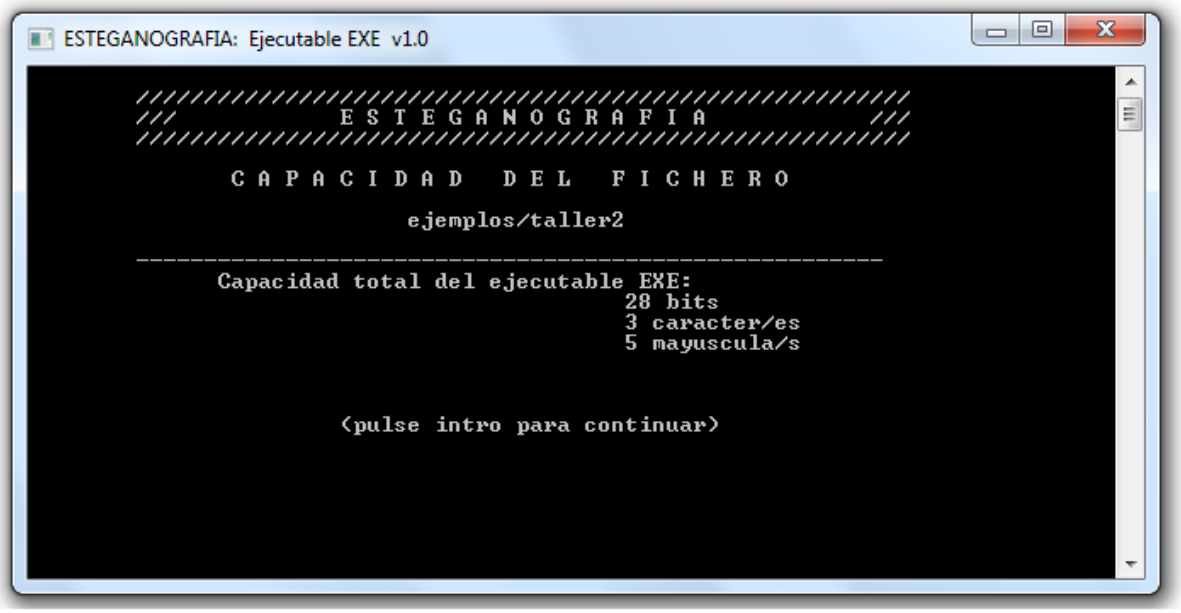


Ilustración 59: pantalla de cálculo de capacidad en ejecutable EXE

PANTALLA DE ESTEGANOGRAFIADO EN EJECUTABLE EXE:

Pantalla que muestra las dos opciones disponibles para esteganografiar un mensaje en un ejecutable EXE obtenido del código C original. Las opciones dependen de la capacidad que se desea obtener, el código siempre tiene una capacidad menor para caracteres que para mayúsculas. El programa incluirá en el mensaje a esteganografiar un bit para indicar la opción de esteganografiado elegida.

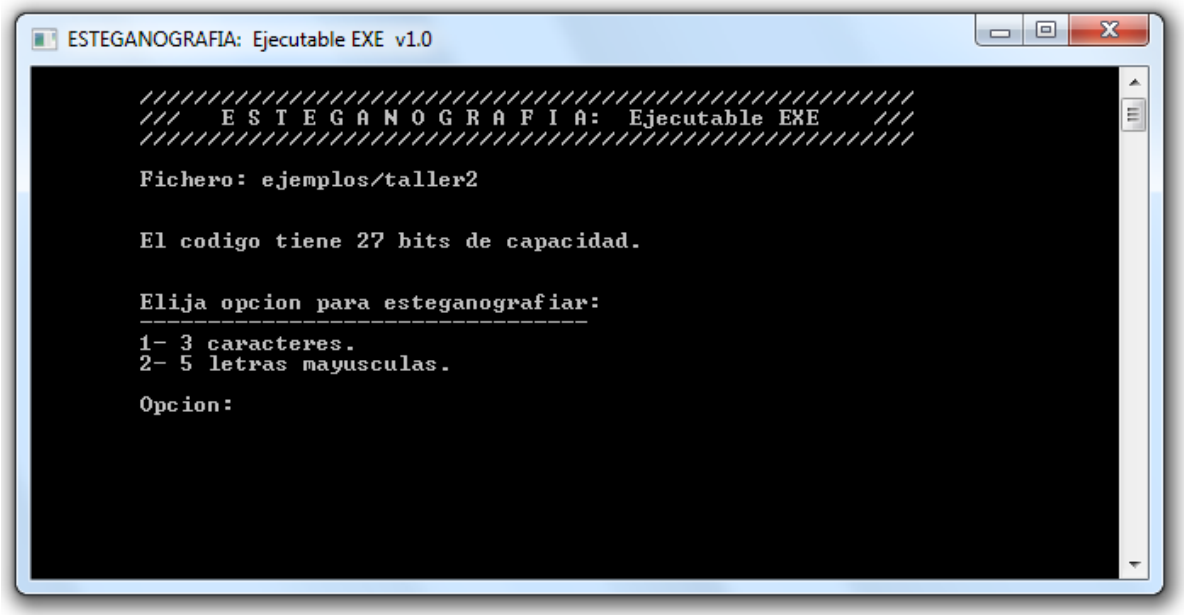


Ilustración 60: Pantalla de esteganografiado en ejecutable EXE

Opción 1:

Esteganografiado de caracteres:

Al inicio de la cadena de bits a esteganografiar se incluye un cero para indicar esta opción.

OPCIÓN 1

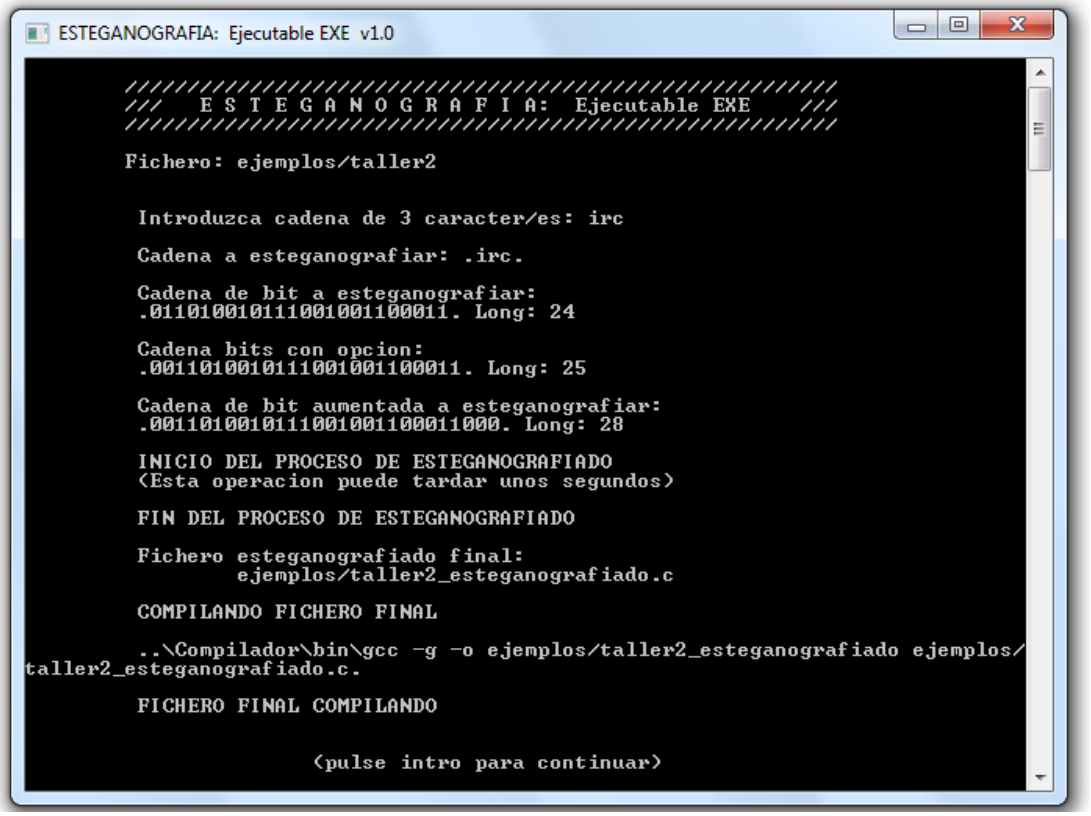


Ilustración 61: Pantalla de esteganografiado de caracteres en código C

Opción 2:

Esteganografiado de mayúsculas:

Al inicio de la cadena de bits a esteganografiar se incluye un uno para indicar esta opción.

OPCIÓN 2

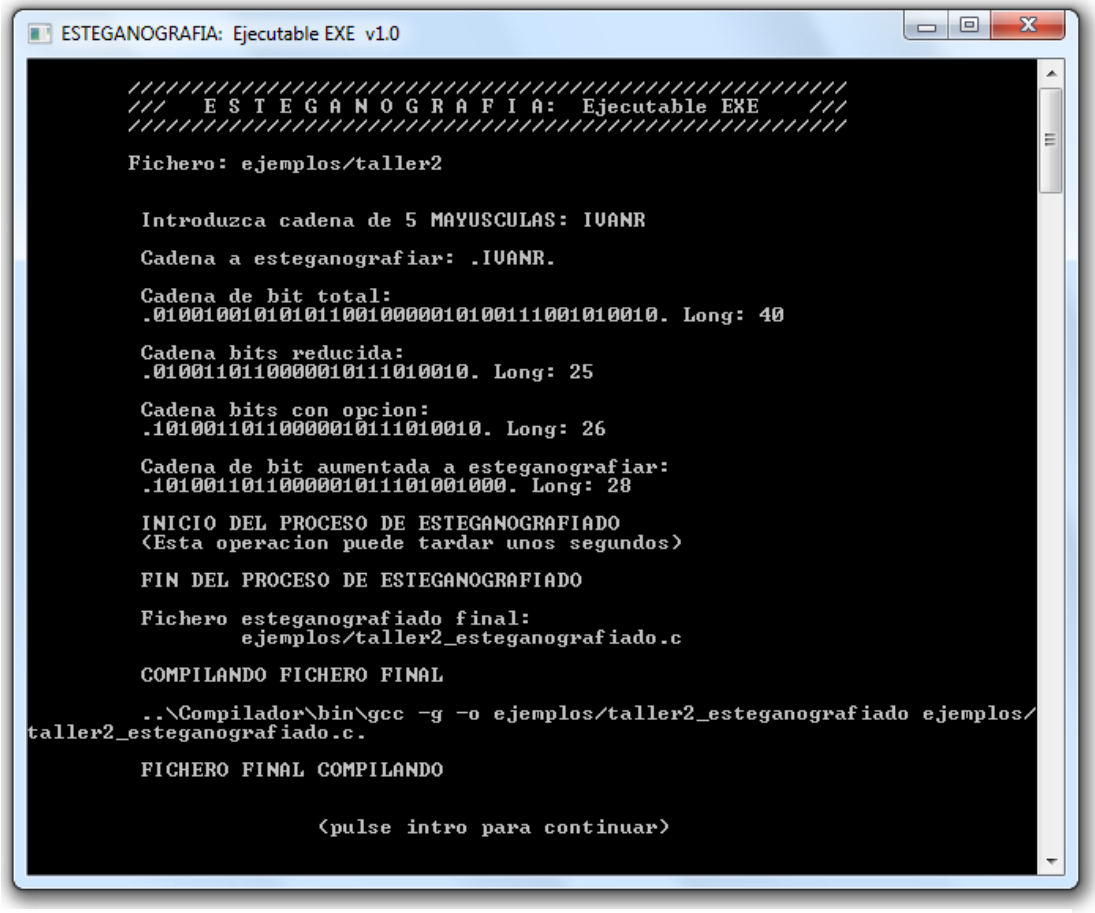


Ilustración 62: Pantalla de esteganografiado de mayúsculas en código C

PANTALLA DE DESESTEGANOGRAFIADO EN EJECUTABLE EXE:

Pantalla que muestra el mensaje desesteganografiado del ejecutable EXE que se obtuvo al esteganografiar el código C original. El programa es capaz de reconocer la opción de esteganografiado que se utilizó por lo que no es necesario indicárselo en este paso, para ello utiliza el llamado "bit de control".

En la Ilustración 63 podemos observar como el primer bit desesteganografiado es un cero, lo que le indica al programa que la opción que se utilizó para el esteganografiado fue la de caracteres. Con lo que el programa únicamente tiene que transformar la cadena en binario obtenida a su equivalente en caracteres para obtener el mensaje esteganografiado.

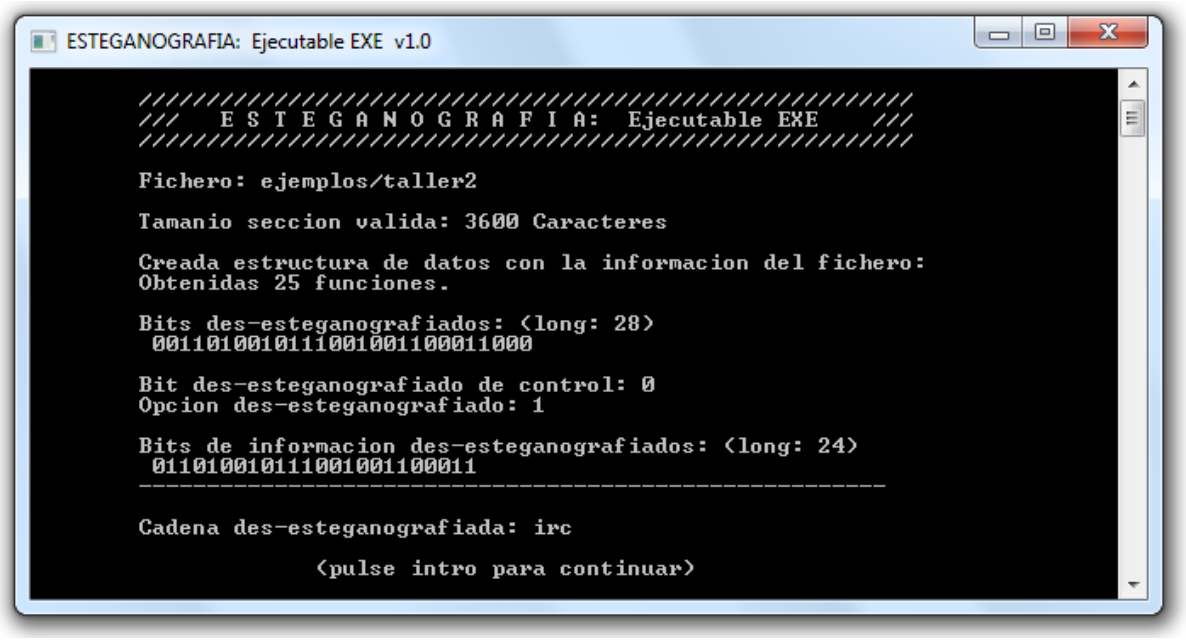


Ilustración 63: Pantalla de desesteganografiado de caracteres en código C

Ahora, en la Ilustración 64 podemos observar el caso contrario, el primer bit desesteganografiado es un uno, lo que le indica al programa que la opción que se utilizó para el esteganografiado fue la de mayúsculas. Con lo que el programa primero tiene que ampliar la cadena reducida en binario obtenida y posteriormente transformarla a una a su equivalente en caracteres para obtener el mensaje esteganografiado.

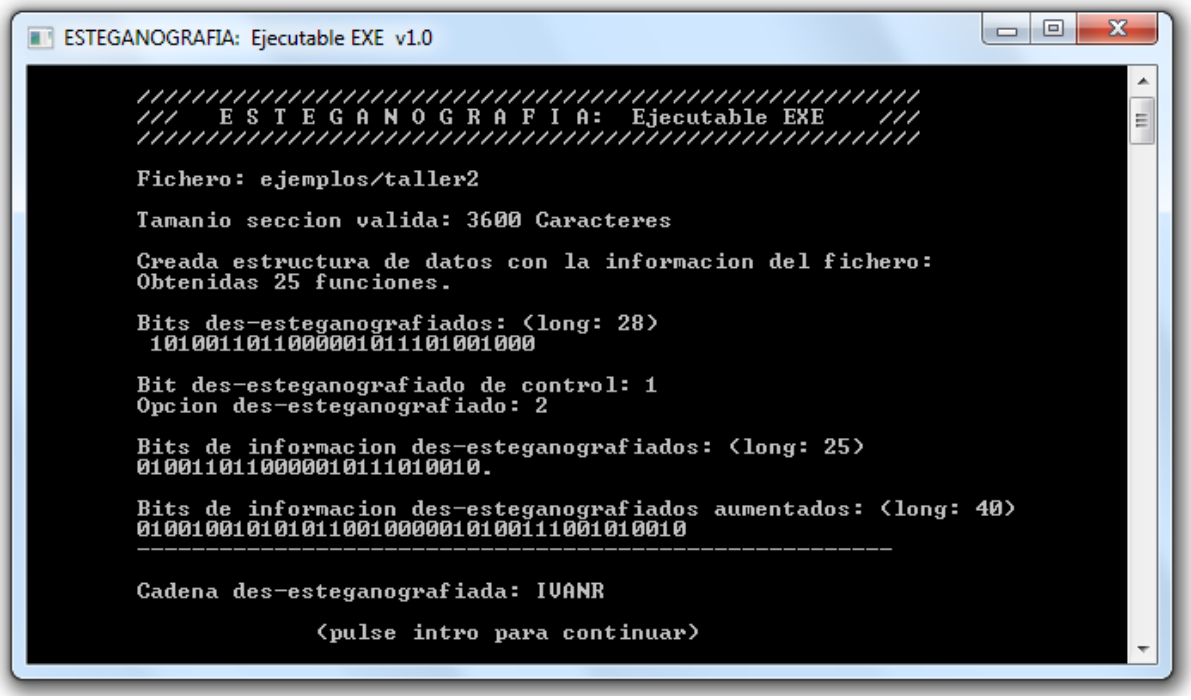


Ilustración 64: Pantalla de desesteganografiado de caracteres en código C

PANTALLA DE MENÚ DE ESTEGANOGRAFIADO EN CÓDIGO C AMPLIADO:

Pantalla que muestra las distintas opciones para esteganografiar con el método ampliado un fichero de código C usando como Estego-objeto el código C original.

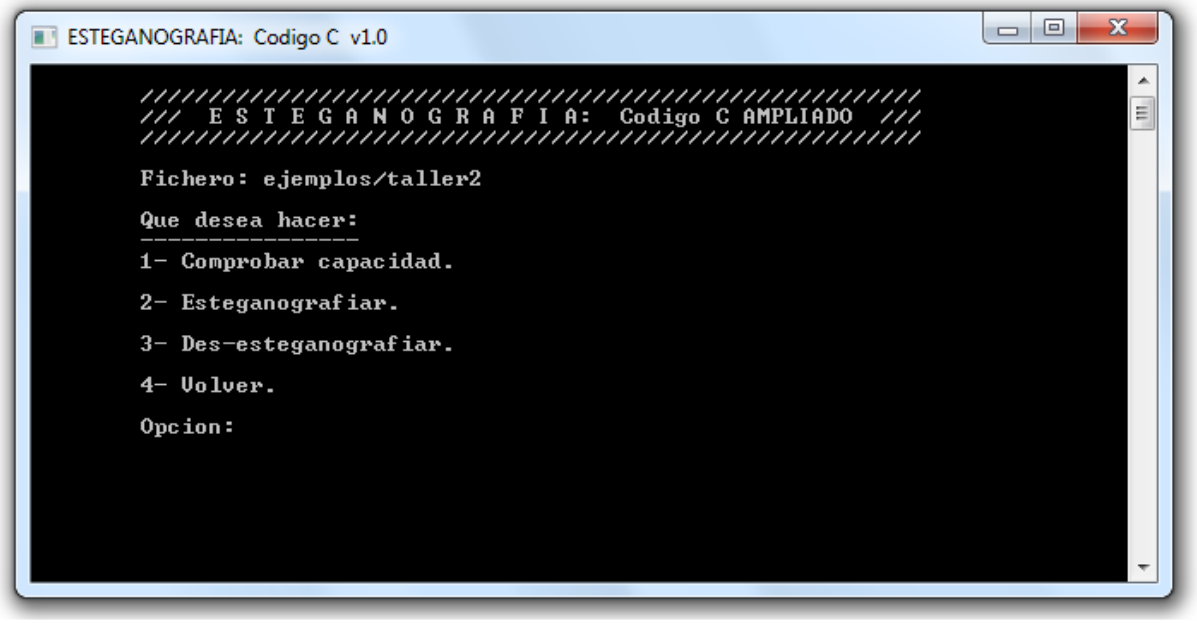


Ilustración 68: Pantalla de menú de esteganografiado en código C ampliado

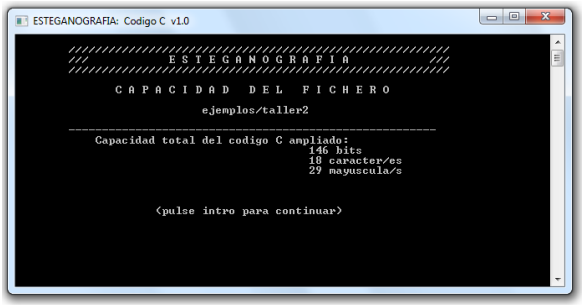
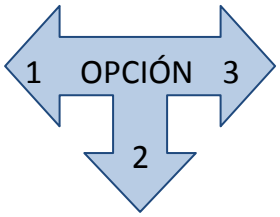


Ilustración 67: Miniatura pantalla de cálculo de capacidad en código C ampliado

(Página 72)



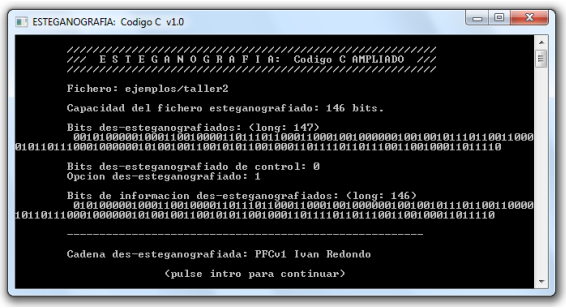


Ilustración 66: Miniatura pantalla de desesteganografiado en código C ampliado

(Página 74)

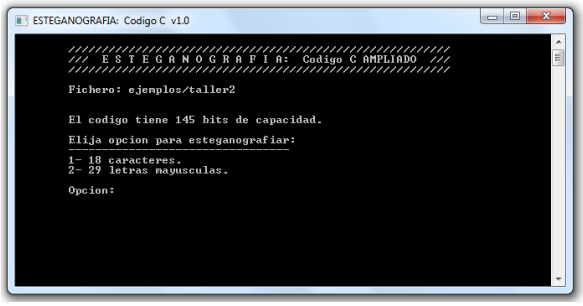


Ilustración 65: Miniatura pantalla de esteganografiado en código C ampliado

(Página 72)

PANTALLA DE CAPACIDAD TOTAL EN CÓDIGO C AMPLIADO:

Pantalla que muestra la capacidad de esteganografiado que tiene el fichero de código C con el modo de esteganografiado combinado de operaciones matemáticas y de funciones y parámetros explicado en el punto 3.3. ALGORITMO DE ESTEGANOGRAFIADO AMPLIADO EN FICHEROS DE CÓDIGO .C del documento.

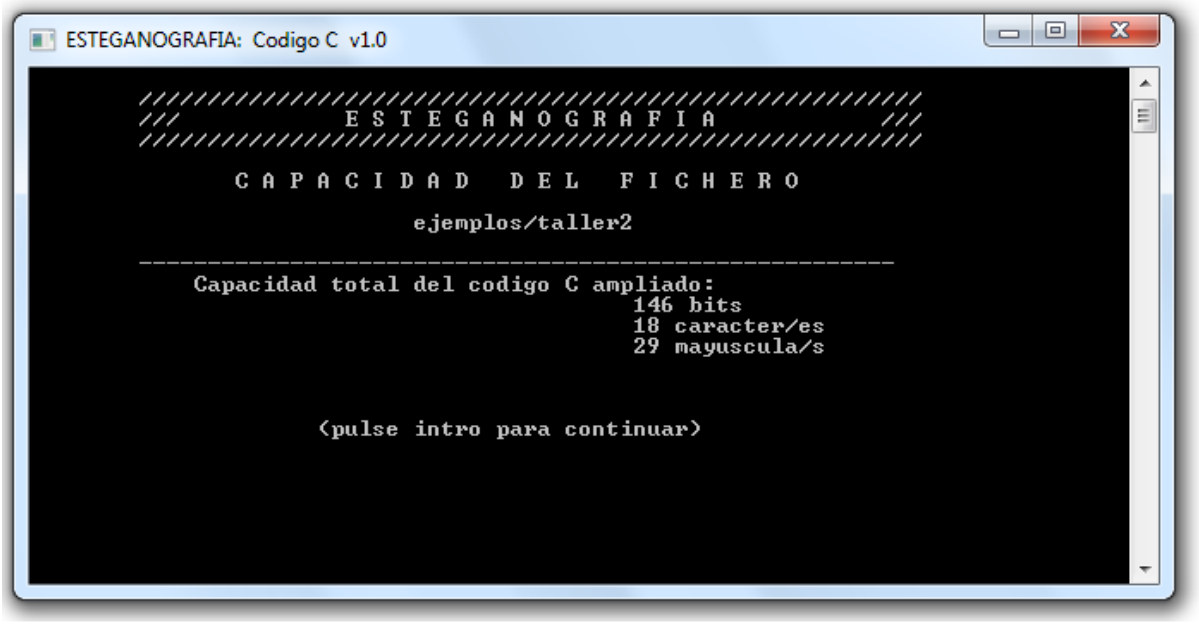


Ilustración 69: Pantalla de cálculo de capacidad en código C ampliado

PANTALLA DE ESTEGANOGRAFIADO EN CÓDIGO C AMPLIADO:

Pantalla que muestra las dos opciones disponibles para esteganografiar un mensaje en un fichero de código C usando el método ampliado y como Estego-objeto el código C original. Las opciones dependen de la capacidad que se desea obtener, el código siempre tiene una capacidad menor para caracteres que para mayúsculas. El programa incluirá en el mensaje a esteganografiar un bit para indicar la opción de esteganografiado elegida.

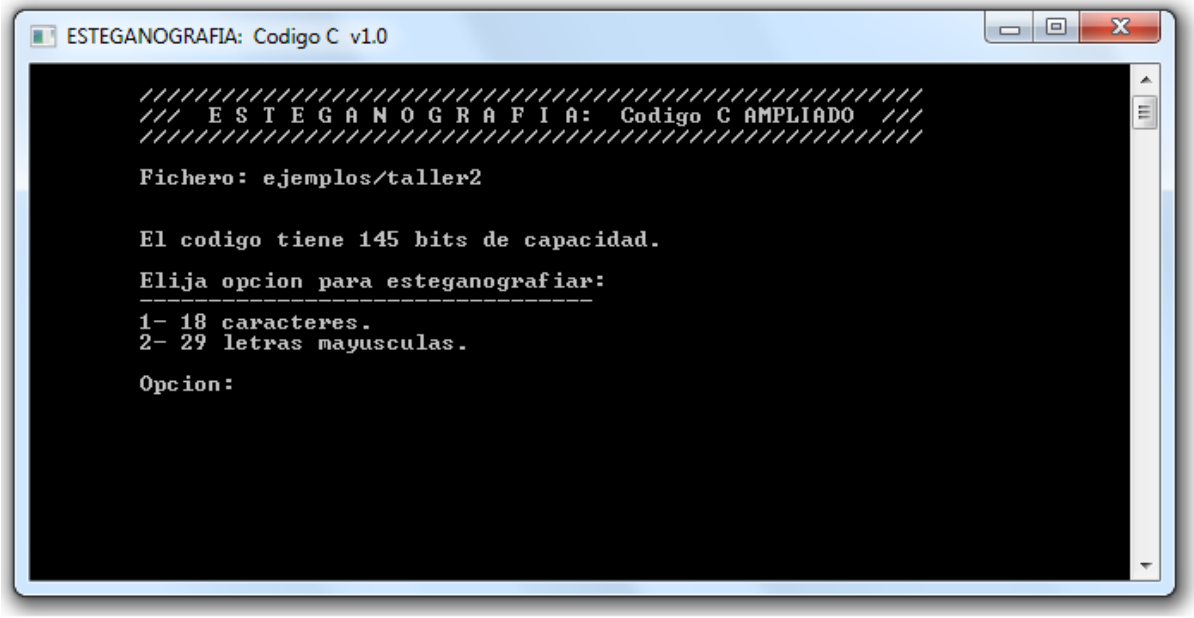


Ilustración 70: Pantalla de esteganografiado en código C ampliado

Opción 1:

Esteganografiado de caracteres:

Al inicio de la cadena de bits a esteganografiar se incluye un cero para indicar esta opción.

OPCIÓN 1

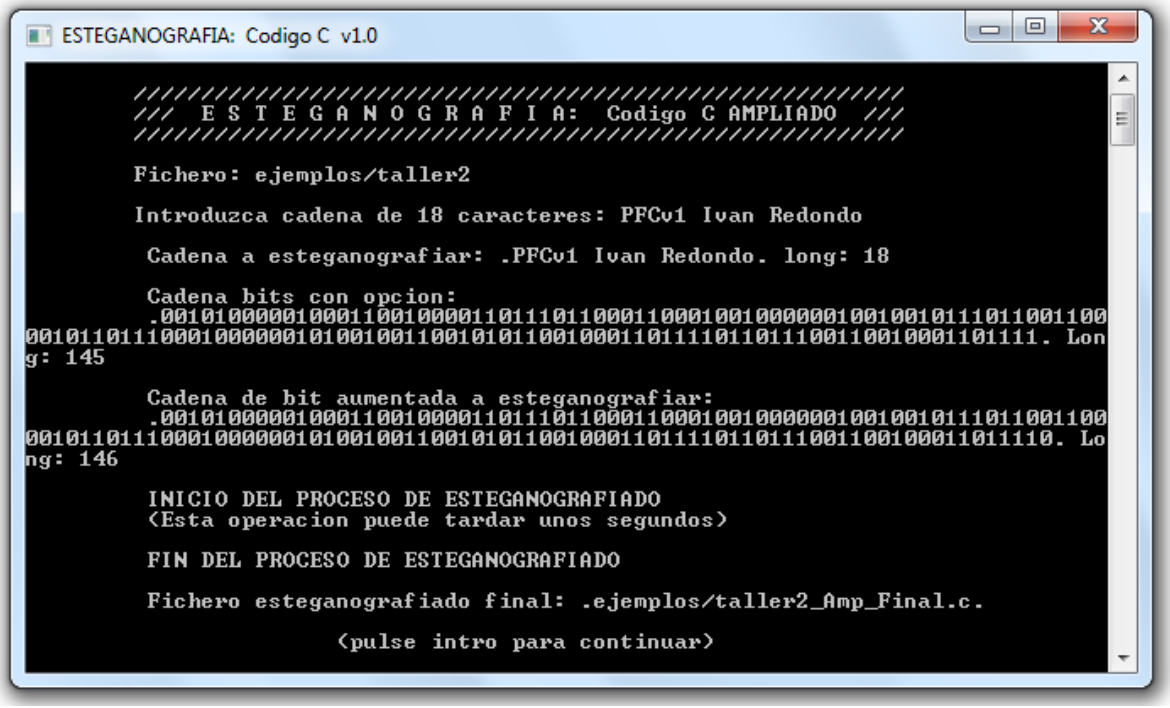


Ilustración 71: Pantalla de esteganografiado de caracteres en código C ampliado

Opción 2:

Esteganografiado de mayúsculas:

Al inicio de la cadena de bits a esteganografiar se incluye un uno para indicar esta opción.

OPCIÓN 2

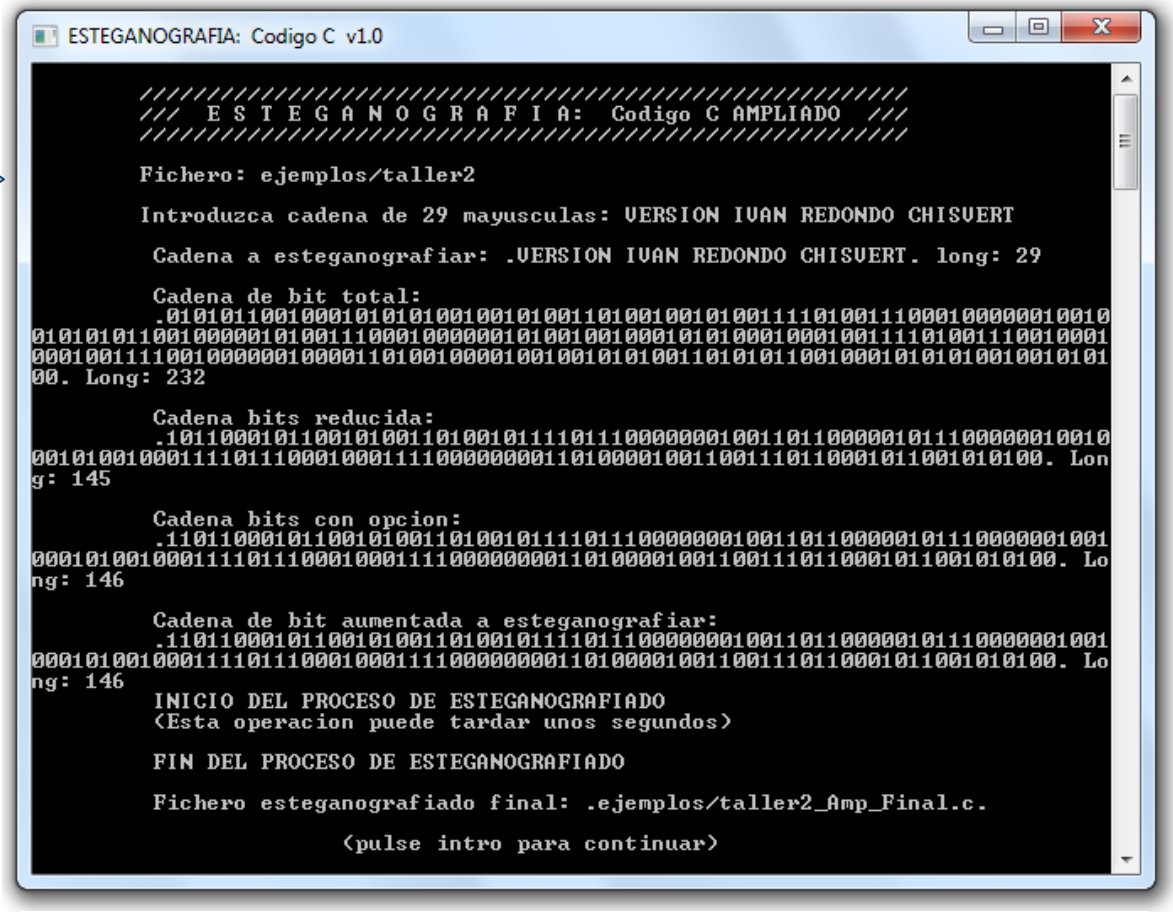


Ilustración 72: Pantalla de esteganografiado de mayúsculas en código C ampliado

PANTALLA DE DESESTEGANOGRAFIADO EN CÓDIGO C AMPLIADO:

Pantalla que muestra el mensaje desesteganografiado del código C original usado como Estego-objeto. El programa es capaz de reconocer la opción de esteganografiado que se utilizó por lo que no es necesario indicárselo en este paso, para ello utiliza el llamado "bit de control".

En la Ilustración 73 podemos observar como el primer bit desesteganografiado es un cero, lo que le indica al programa que la opción que se utilizó para el esteganografiado fue la de caracteres. Con lo que el programa únicamente tiene que transformar la cadena en binario obtenida a su equivalente en caracteres para obtener el mensaje esteganografiado.

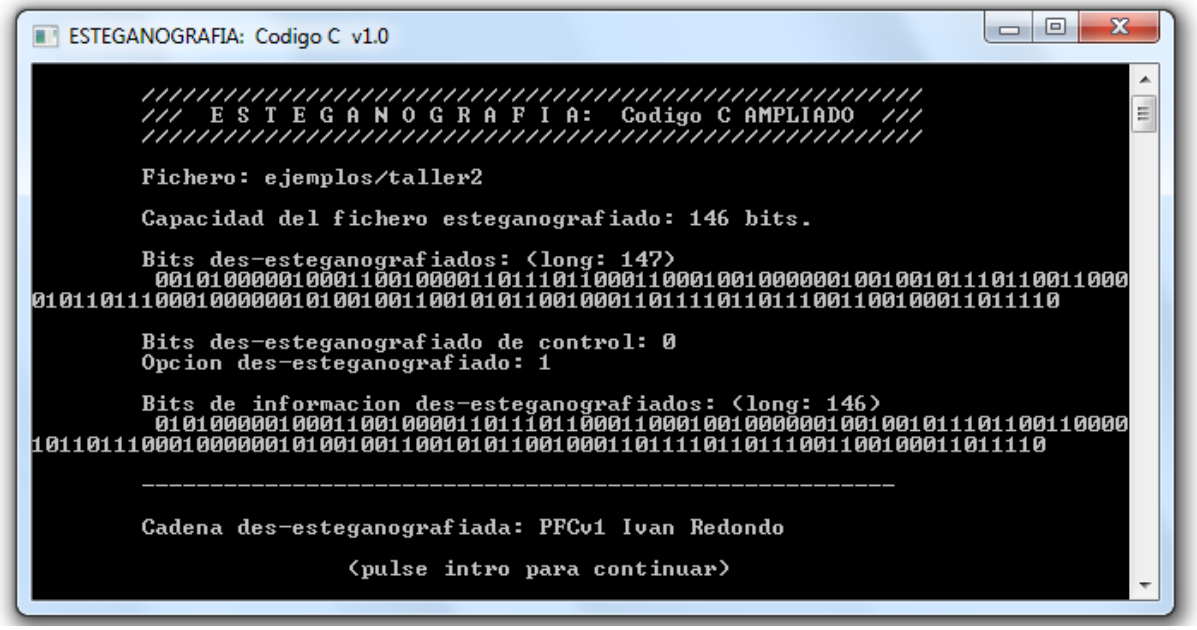


Ilustración 73: Pantalla de desesteganografiado de caracteres en código C

Ahora, en la Ilustración 74 podemos observar el caso contrario, el primer bit desesteganografiado es un uno, lo que le indica al programa que la opción que se utilizó para el esteganografiado fue la de mayúsculas. Con lo que el programa primero tiene que ampliar la cadena reducida en binario obtenida y posteriormente transformarla a una a su equivalente en caracteres para obtener el mensaje esteganografiado.

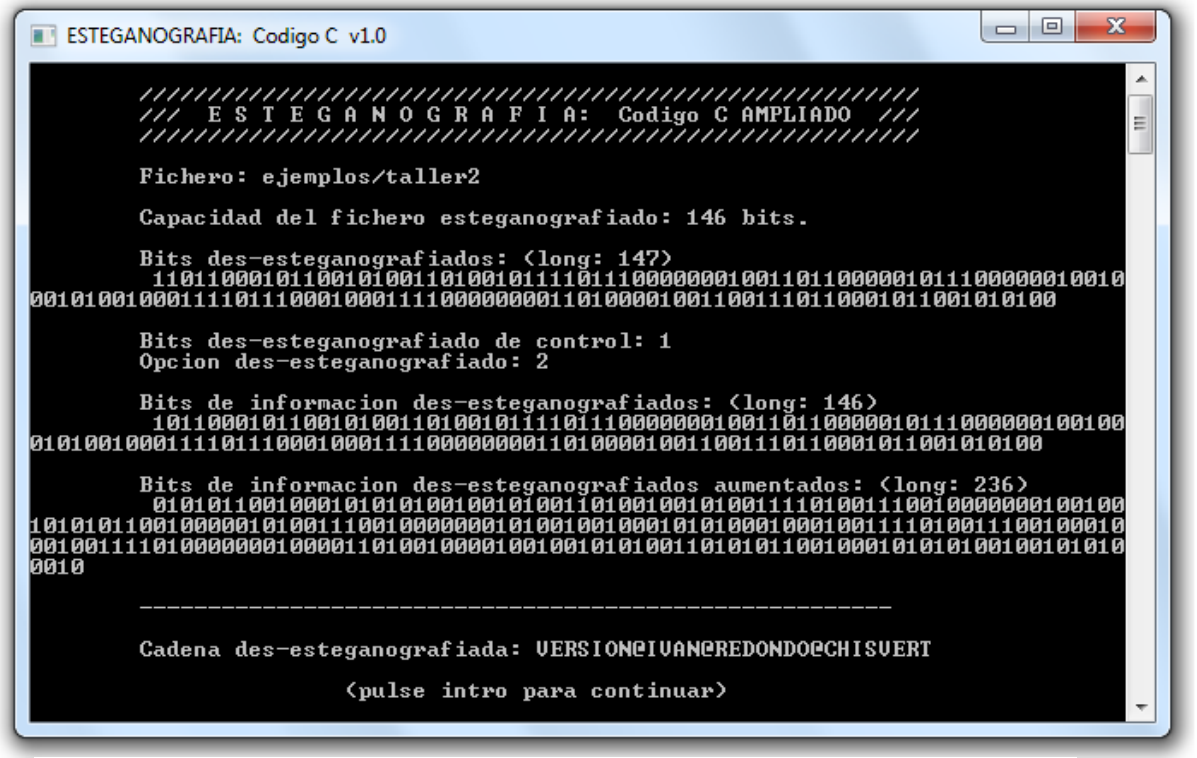


Ilustración 74: Pantalla de desesteganografiado de caracteres en código C

5.3. METODOLOGÍA Y PRUEBAS FUNCIONALES DEL PROGRAMA

Como este proyecto está más enfocado a la investigación que a un simple desarrollo de software, no se disponía de una especificación de requisitos totalmente detallados antes de poder iniciar la etapa de diseño. Debido a esto, se optó por seguir una metodología de desarrollo evolutivo que permitiese que los requisitos no estuvieran totalmente especificados antes de comenzar el desarrollo.

Como ya hemos dicho el modelo en cascada requiere tener una especificación de requerimientos totalmente detallada para poder iniciar la etapa del diseño. Si a lo largo del proceso de desarrollo se cambian los requisitos hay que rehacer parte del trabajo de diseño e implementación para poder hacer frente a los nuevos cambios.

El siguiente diagrama muestra el modelo de desarrollo evolutivo seguido, el modelo iterativo incremental.

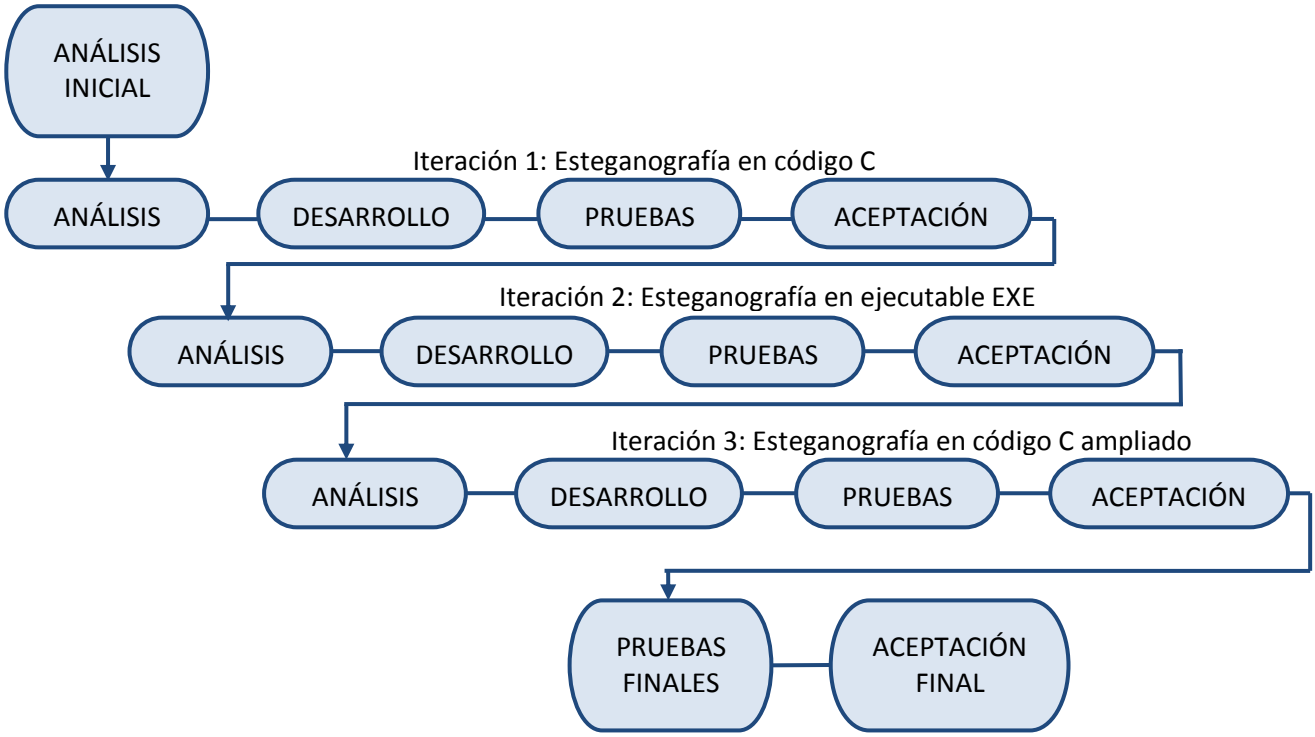


Diagrama del modelo iterativo incremental

Al seguir este modelo, en un primer análisis se definió como iba a ser el proyecto básico, y a medida que la investigación y el software desarrollado iban avanzando, se encontraban nuevos métodos de esteganografiado que se iban incluyendo en las siguientes iteraciones. Una vez completadas todas ellas, se revisaban en su contexto final antes de la aceptación definitiva del software como conjunto único de todas ellas.

Esta metodología fue la seleccionada debido a que con cada interacción se obtenía un software funcional que se podía utilizar para investigar los siguientes nuevos métodos de esteganografiado.

Como cada iteración sigue a su vez el típico modelo en cascada, al inicio de dicha iteración sí que se conocía cual era el alcance o la funcionalidad que se pretendía desarrollar en esa interacción por lo que cada una de ellas para poder ser aceptadas debía de pasar una serie de pruebas funcionales antes de poder comenzar la siguiente interacción.

Definiendo una prueba funcional como una prueba basada en la ejecución y análisis de las funcionalidades previamente diseñadas para el software. Estas pruebas funcionales realizadas buscan evaluar cada una de las partes con las que cuenta el programa de forma específica, concreta y exhaustiva para probar y validar su correcto funcionamiento.

Para este software, se tomara el roll de un usuario final y se realizara un plan de pruebas que consistirá en una serie de pruebas funcionales manuales que aun no existiendo una lista de requisitos que se han de cumplir, servirán para garantizar que el programa hace lo que debe y que no falla para cada una de las partes que lo forman.

PLAN DE PRUEBAS GENERAL		
ITERACIÓN	**PRUEBAS**	**RESULTADO**
1- Pruebas Esteganografía en código C	Prueba 2: Solicitud de cadena de caracteres Prueba 3: Solicitud de cadena de mayúsculas	Correctas
2- Pruebas Esteganografía en ejecutable EXE	Prueba 4: Solicitud de cadena de caracteres Prueba 5: Solicitud de cadena de mayúsculas	Correctas
3- Pruebas Esteganografía en código C ampliado	Prueba 6: Solicitud de cadena de caracteres Prueba 7: Solicitud de cadena de mayúsculas	Correctas
4- Pruebas Finales	Prueba 1: Solicitud de fichero	Correcta

Tabla 21: Plan de pruebas general

PLAN DE PRUEBAS

IDENTIFICADOR	PRUEBA 1	
NOMBRE	SOLICITUD DE FICHERO	
PROPÓSITO	El programa tiene que ser capaz de encontrar el fichero que se le indica, validar que es correcto y continuar la ejecución. En caso de que el fichero no se encuentre o no sea correcto, se informara de ello al usuario y se volverá a pedir un fichero, no deteniendo la ejecución.	
CASOS DE PRUEBA	SALIDA	RESULTADO
Fichero correcto	El programa reconoce el archivo de entrada como correcto.	Correcto
Fichero incorrecto	El programa reconoce el archivo de entrada como incorrecto.	Correcto

Tabla 22: Prueba solicitud de fichero

FICHERO CORRECTO:

Ilustración 75: Prueba de fichero OK

FICHERO INCORRECTO:

Ilustración 76: Prueba de fichero KO

IDENTIFICADOR	PRUEBA 2	
NOMBRE	SOLICITUD CADENA CARACTERES PARA CÓDIGO C	
PROPÓSITO	El programa ha de ser flexible a la hora de solicitar la cadena de caracteres a esteganografiar. El programa no ha de fallar o funcionar mal ante cualquier tipo de longitud de entrada. Siendo a elección del usuario el utilizar toda la capacidad disponible o no.	
CASOS DE PRUEBA	SALIDA	RESULTADO
Longitud exacta	El programa utiliza para el esteganografiado la cadena introducida.	Correcto
Longitud inferior	El programa utiliza para el esteganografiado la cadena introducida y rellena con espacios en blanco hasta la longitud pedida.	Correcto
Longitud superior	El programa utiliza para el esteganografiado la cadena introducida truncándola a la longitud pedida.	Correcto
Longitud cero	El programa actúa como si la cadena fuera de longitud menor y rellena con espacios en blanco hasta la longitud pedida en este caso la longitud total.	Correcto

Tabla 23: Prueba solicitud cadena caracteres para código c

PANTALLA DE SOLICITUD DE CADENA PARA EL ESTEGANOGRAFIADO DE CARACTERES EN CÓDIGO C:

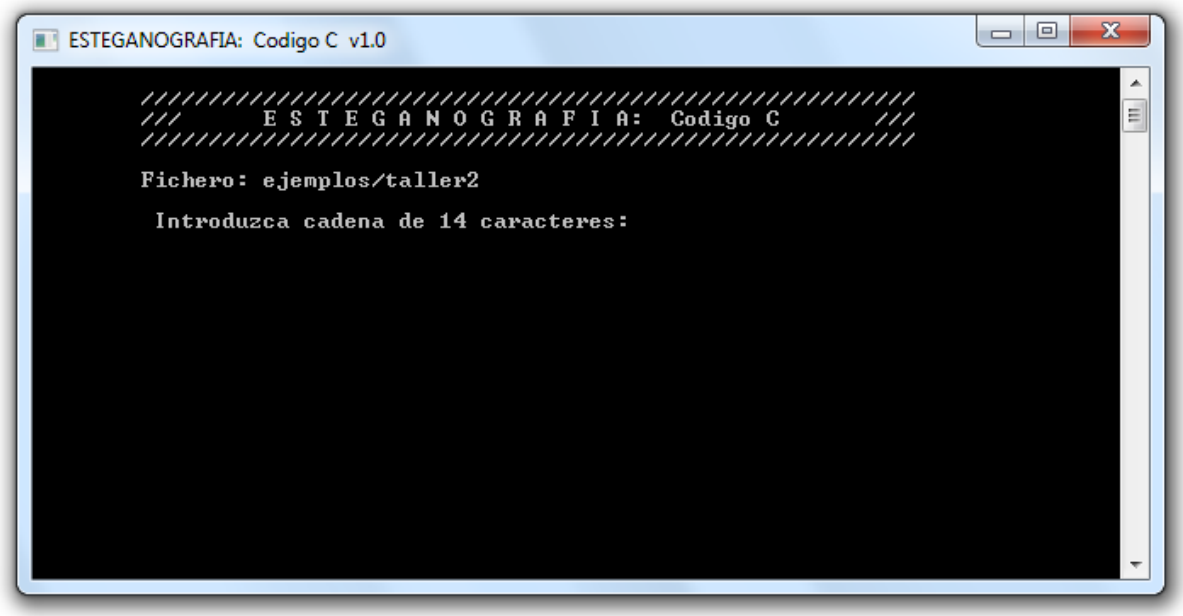


Ilustración 77: Pantalla de solicitud de cadena de caracteres para código C

LONGITUD EXACTA:

El programa solicita una cadena de 14 caracteres, se introduce la cadena “Ivan Redondo 1” con una longitud exacta de 14 caracteres, el programa la lee y la utiliza sin problemas.

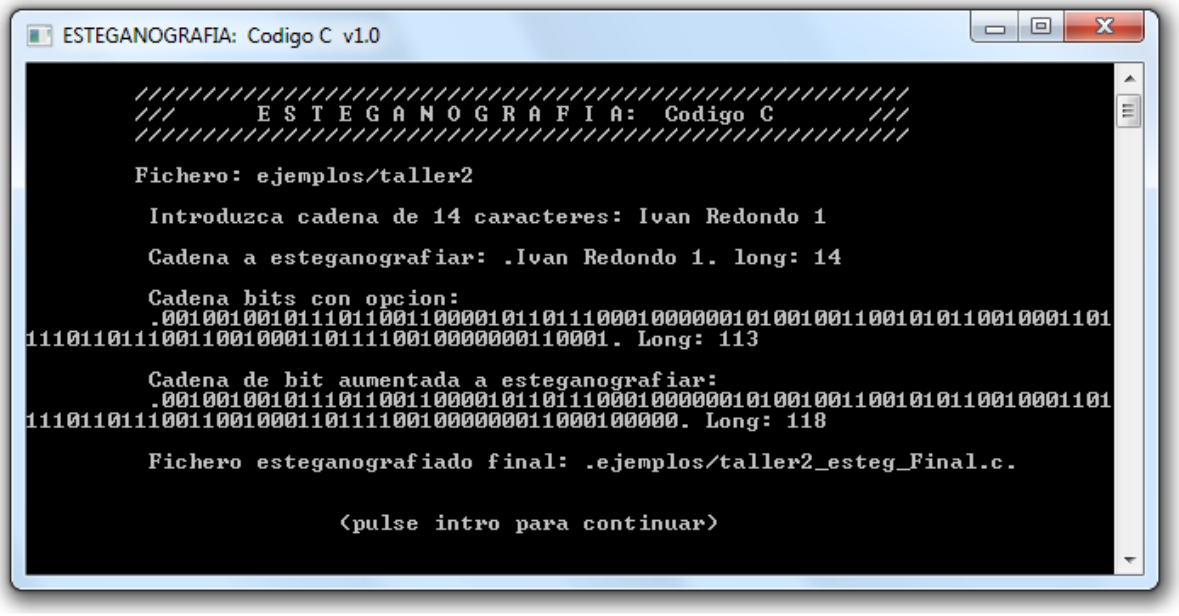


Ilustración 78: Prueba de esteganografiado longitud exacta para caracteres en código C

Para verificar que el programa ha utilizado la cadena introducida sin problemas desesteganografiamos para obtener de nuevo la cadena y observamos que esta sigue siendo la cadena “Ivan Redondo 1” con una longitud exacta de 14 caracteres.

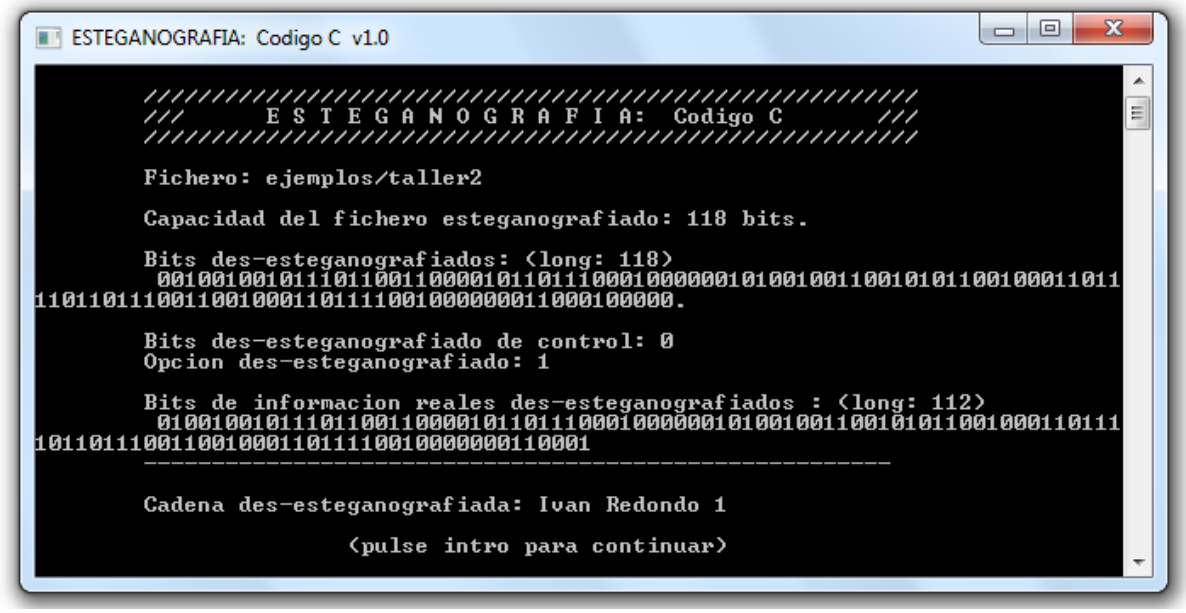


Ilustración 79: Prueba de desesteganografiado longitud exacta para caracteres en código C

LONGITUD INFERIOR:

El programa solicita una cadena de 14 caracteres, se introduce la cadena “IvanRCH” con una longitud inferior, en este caso de solo 7 caracteres, el programa la lee, detecta que su longitud es inferior, la aumenta concatenándola 7 espacios en blanco y la utiliza sin problemas.

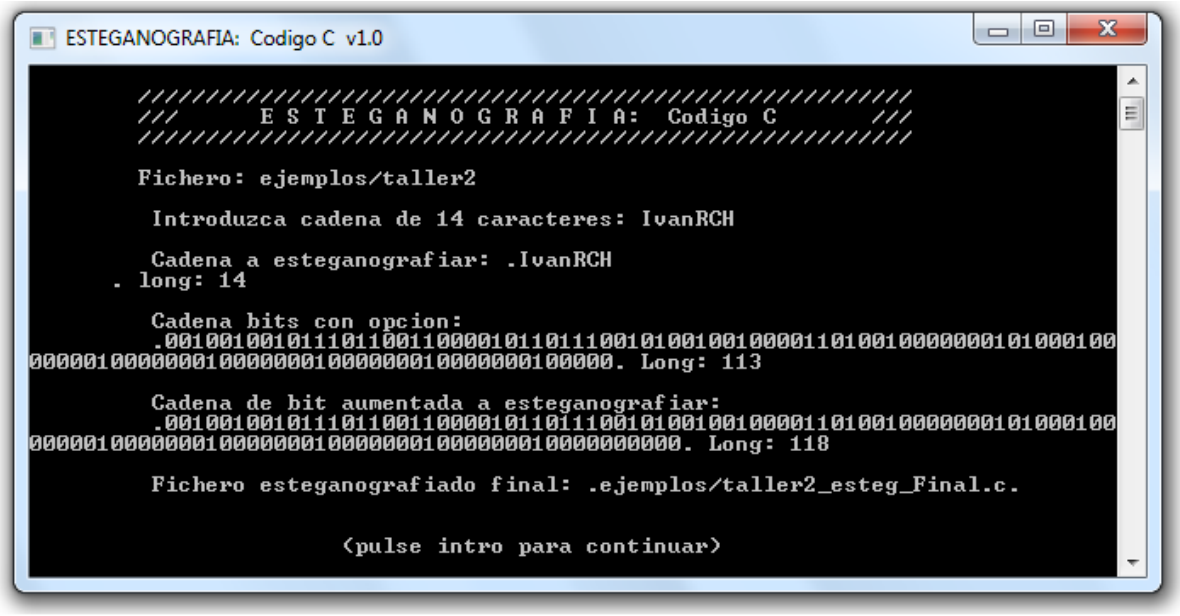


Ilustración 80: Prueba de esteganografiado longitud inferior para caracteres en código C

LONGITUD SUPERIOR:

El programa solicita una cadena de 14 caracteres, se introduce la cadena “Ivan Redondo Chisvert” con una longitud superior, en este caso 21 caracteres, el programa la lee, detecta que su longitud es superior, la trunca a la longitud requerida de 14 caracteres y la utiliza sin problemas.

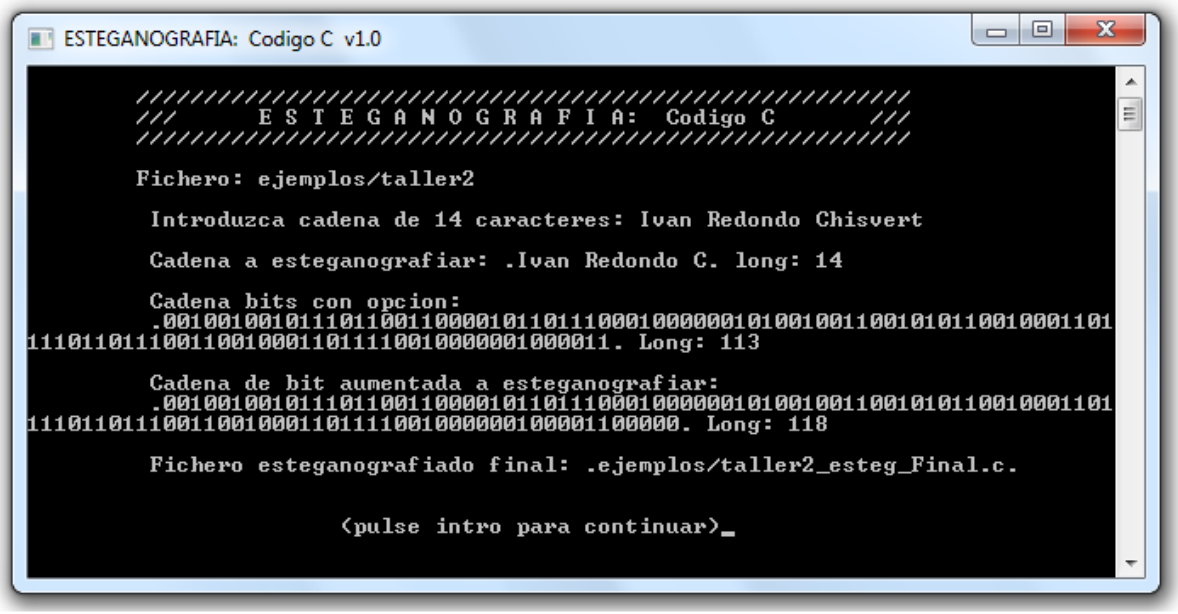


Ilustración 81: Prueba de esteganografiado longitud superior para caracteres en código C

LONGITUD CERO:

El programa solicita una cadena de 14 caracteres y no se introduce cadena alguna, el programa la lee como una cadena vacía y actúa como si de una cadena de longitud inferior se tratara, la aumenta de cero a la longitud requerida concatenándola 14 espacios en blanco y la utiliza sin problemas.

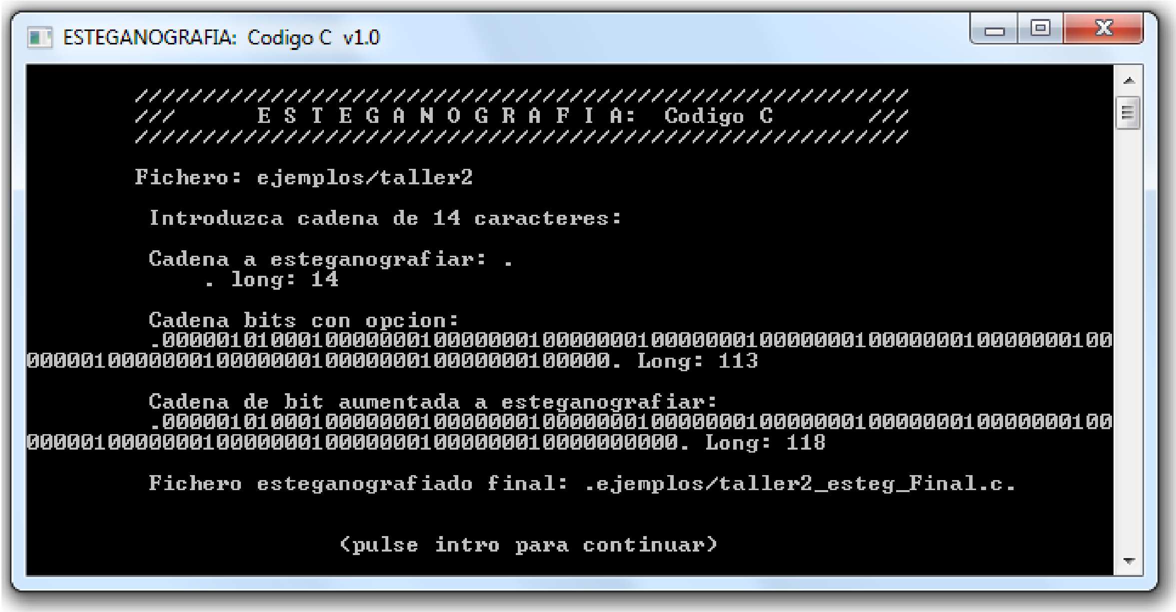


Ilustración 82: Prueba de esteganografiado longitud cero para caracteres en código C

IDENTIFICADOR	PRUEBA 3	
NOMBRE	SOLICITUD CADENA MAYÚSCULAS PARA CÓDIGO C	
PROPÓSITO	El programa ha de ser flexible a la hora de solicitar la cadena de mayúsculas a esteganografiar. El programa no ha de fallar o funcionar mal ante cualquier tipo de longitud de entrada. Siendo a elección del usuario el utilizar toda la capacidad disponible o no.	
CASOS DE PRUEBA	SALIDA	RESULTADO
Longitud exacta	El programa utiliza para el esteganografiado la cadena de mayúsculas introducida.	Correcto
Longitud inferior	El programa utiliza para el esteganografiado la cadena de mayúsculas introducida y rellena con espacios en blanco hasta la longitud pedida.	Correcto
Longitud superior	El programa utiliza para el esteganografiado la cadena de mayúsculas introducida truncándola a la longitud pedida.	Correcto
Longitud cero	El programa actúa como si la cadena fuera de longitud menor y rellena con espacios en blanco hasta la longitud pedida en este caso la longitud total.	Correcto

Tabla 24: Prueba solicitud cadena mayúsculas para código c

PANTALLA DE SOLICITUD DE CADENA PARA EL ESTEGANOGRAFIADO DE MAYÚSCULAS EN CÓDIGO C:

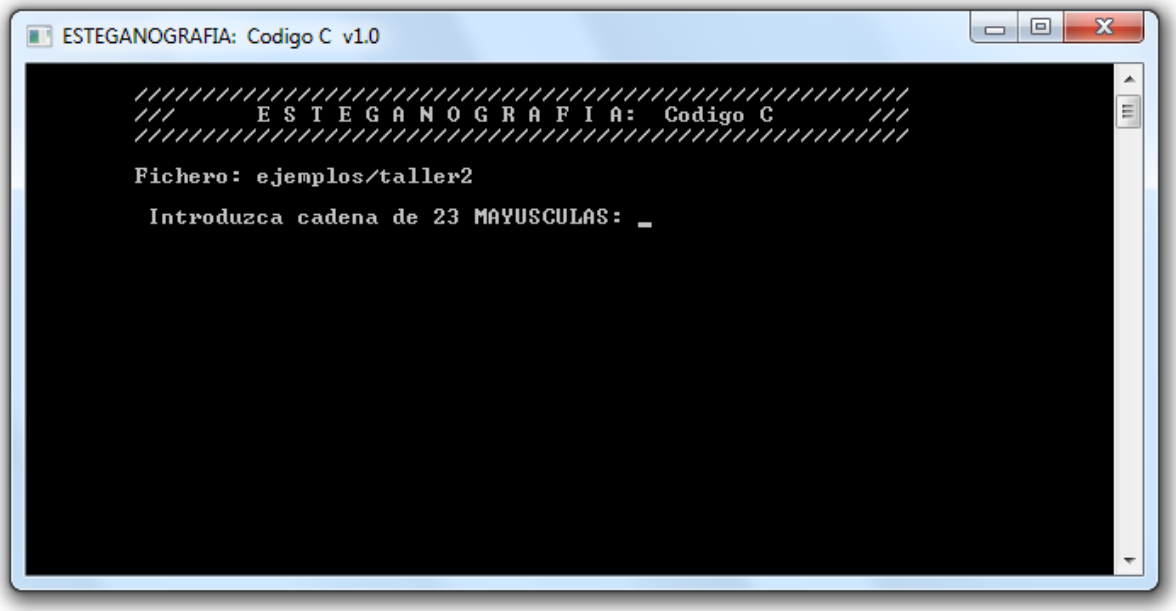


Ilustración 83: Pantalla de solicitud de cadena de mayúsculas para código C

LONGITUD EXACTA:

El programa solicita una cadena de 23 mayúsculas, se introduce la cadena "PFCUNIVERSIDADCARLOSIII" con una longitud exacta de 23 caracteres, el programa la lee y la utiliza sin problemas.

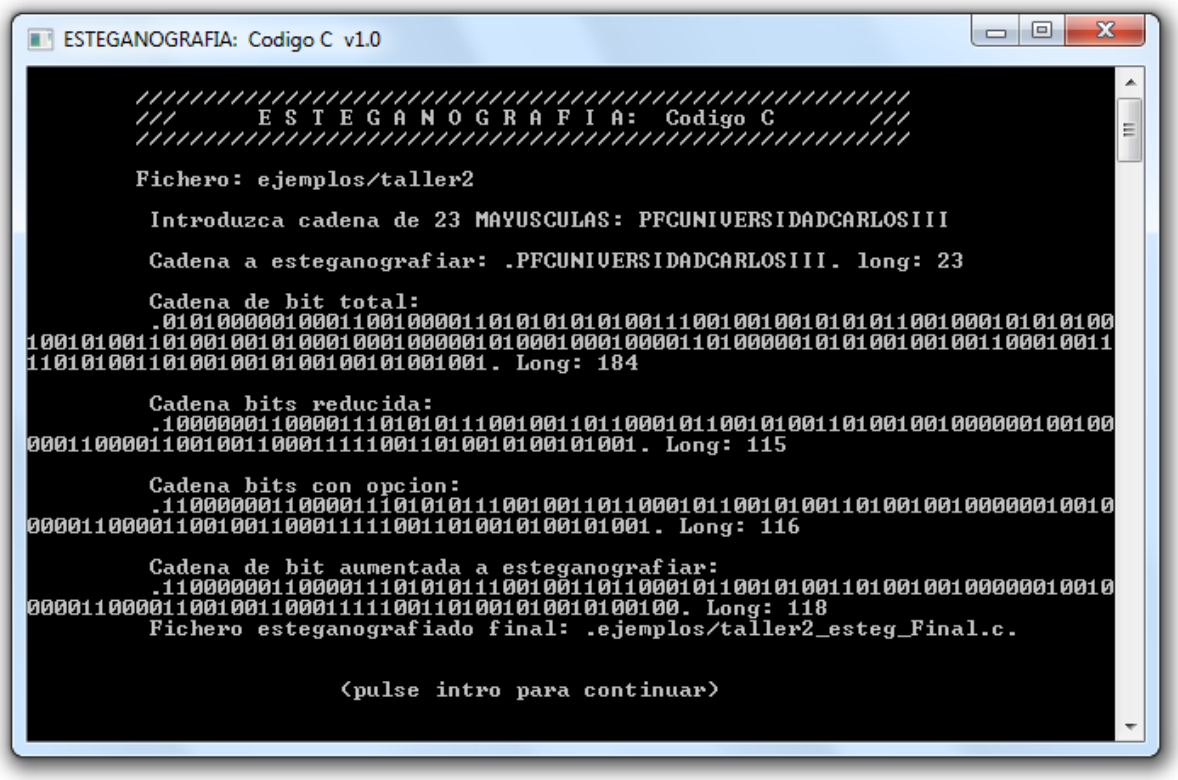


Ilustración 84: Prueba de esteganografiado longitud exacta para mayúsculas en código C

Para verificar que el programa ha utilizado la cadena introducida sin problemas desesteganografiamos para obtener de nuevo la cadena y observamos que esta sigue siendo la cadena "PFCUNIVERSIDADCARLOSIII" con una longitud exacta de 23 caracteres.

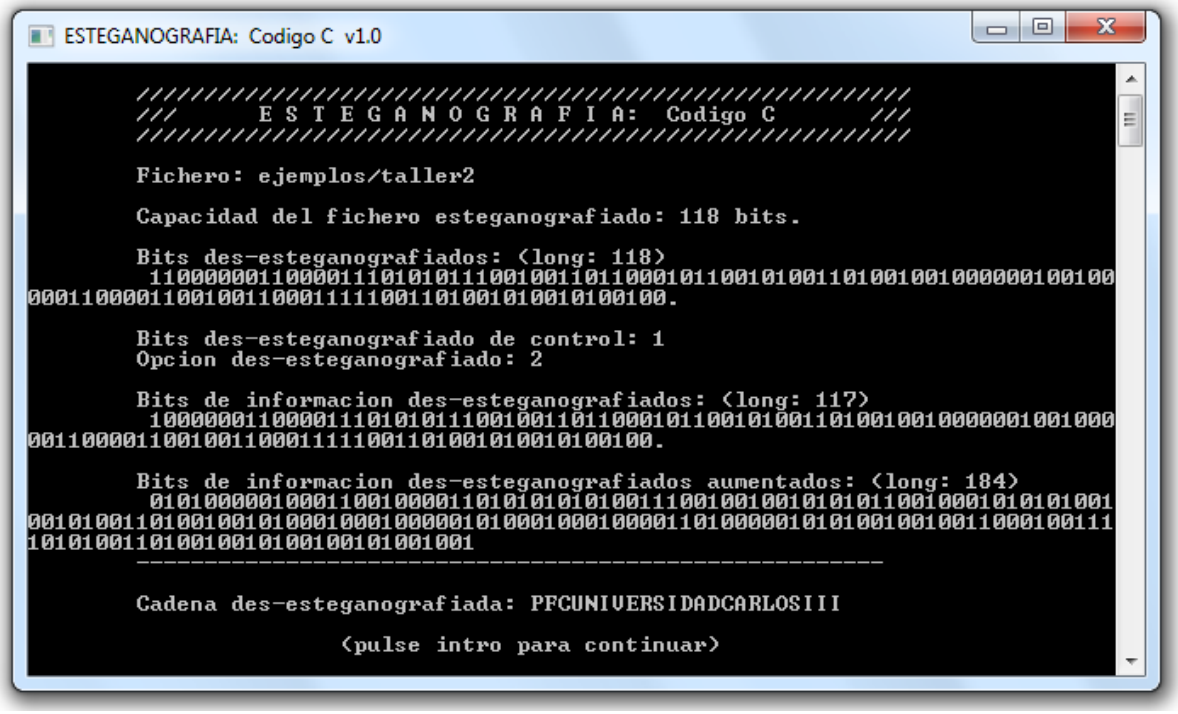


Ilustración 85: Prueba de desesteganografiado longitud exacta para mayúsculas en código C

LONGITUD INFERIOR:

El programa solicita una cadena de 23 mayúsculas, se introduce la cadena "IVANREDONDOCHISVERT" con una longitud inferior, en este caso de solo 19 caracteres, el programa la lee, detecta que su longitud es inferior, la aumenta concatenándola 4 espacios en blanco y la utiliza sin problemas.

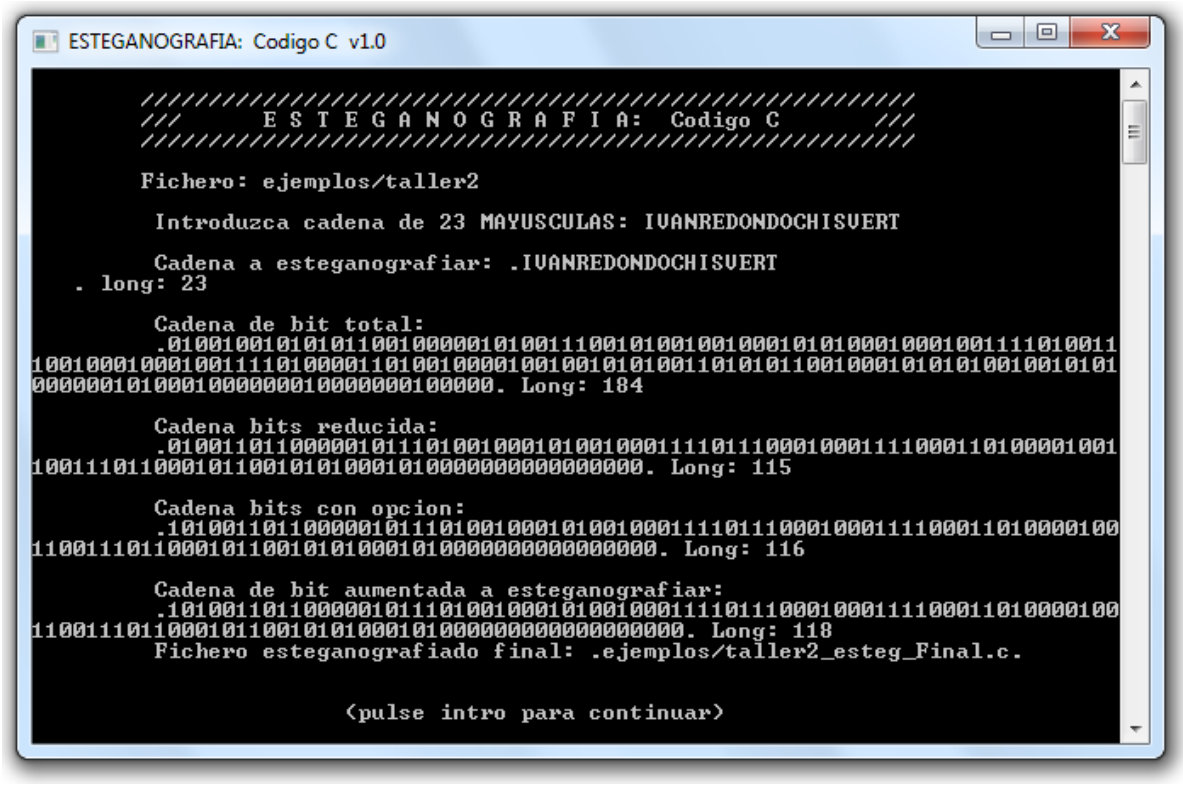


Ilustración 86: Prueba de esteganografiado longitud inferior para mayúsculas en código C

LONGITUD SUPERIOR:

El programa solicita una cadena de 23 mayúsculas, se introduce la cadena "PFCUNIVERSIDADCARLOSTERCERO" con una longitud superior, en este caso 27 caracteres, el programa la lee, detecta que su longitud es superior, la trunca a la longitud requerida de 23 mayúsculas y la utiliza sin problemas.

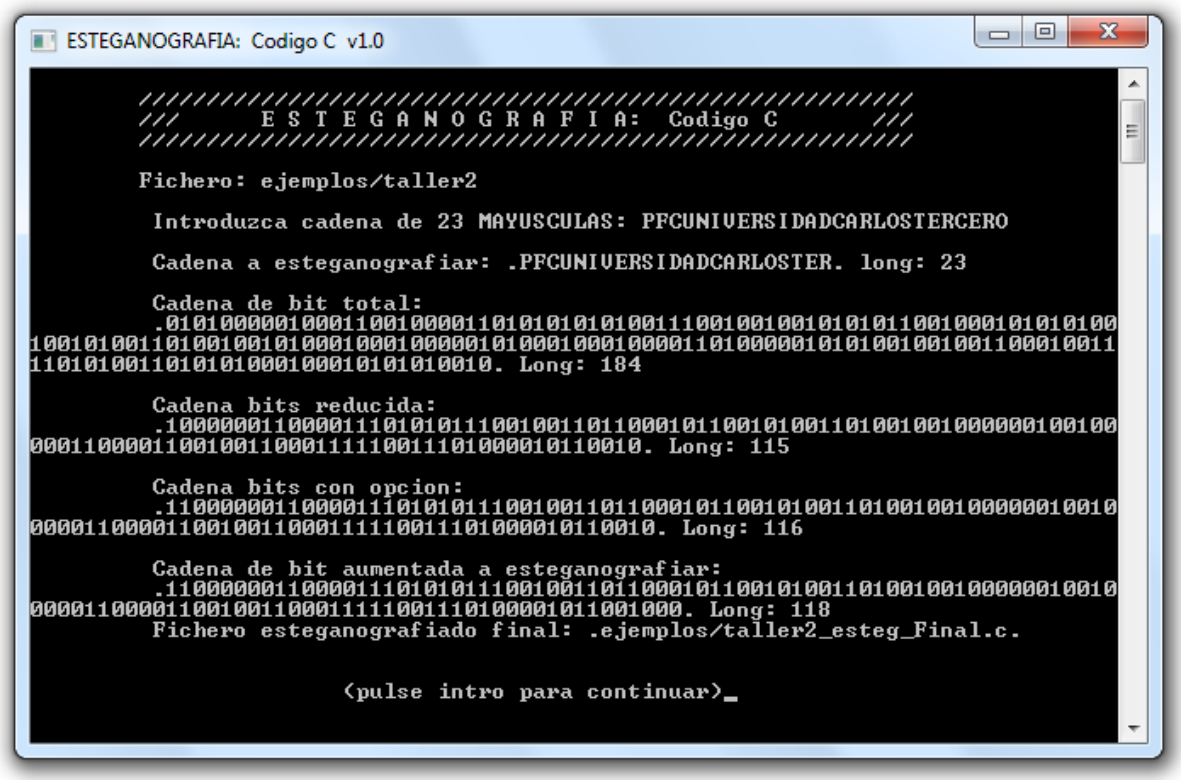


Ilustración 87: Prueba de esteganografiado longitud superior para mayúsculas en código C

LONGITUD CERO:

El programa solicita una cadena de 23 mayúsculas y no se introduce cadena alguna, el programa la lee como una cadena vacía y actúa como si de una cadena de longitud inferior se tratara, la aumenta de cero a la longitud requerida concatenándola 23 espacios en blanco y la utiliza sin problemas.

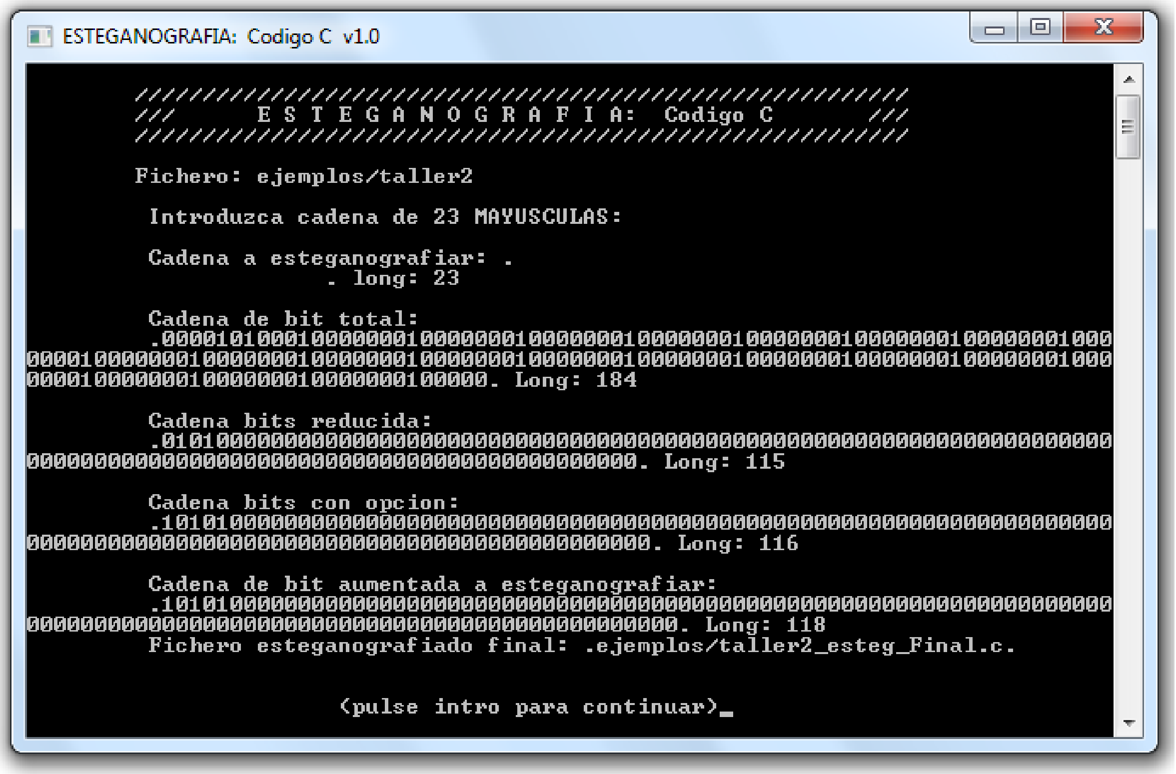


Ilustración 88: Prueba de esteganografiado longitud cero para mayúsculas en código C

IDENTIFICADOR	PRUEBA 4	
NOMBRE	SOLICITUD CADENA CARACTERES PARA EJECUTABLE EXE	
PROPÓSITO	El programa ha de ser flexible a la hora de solicitar la cadena de caracteres a esteganografiar. El programa no ha de fallar o funcionar mal ante cualquier tipo de longitud de entrada. Siendo a elección del usuario el utilizar toda la capacidad disponible o no.	
CASOS DE PRUEBA	SALIDA	RESULTADO
Longitud exacta	El programa utiliza para el esteganografiado la cadena introducida.	Correcto
Longitud inferior	El programa utiliza para el esteganografiado la cadena introducida y rellena con espacios en blanco hasta la longitud pedida.	Correcto
Longitud superior	El programa utiliza para el esteganografiado la cadena introducida truncándola a la longitud pedida.	Correcto
Longitud cero	El programa actúa como si la cadena fuera de longitud menor y rellena con espacios en blanco hasta la longitud pedida en este caso la longitud total.	Correcto

Tabla 25: Prueba solicitud cadena caracteres para ejecutable EXE

PANTALLA DE SOLICITUD DE CADENA PARA EL ESTEGANOGRAFIADO DE CARACTERES EN EJECUTABLE EXE:

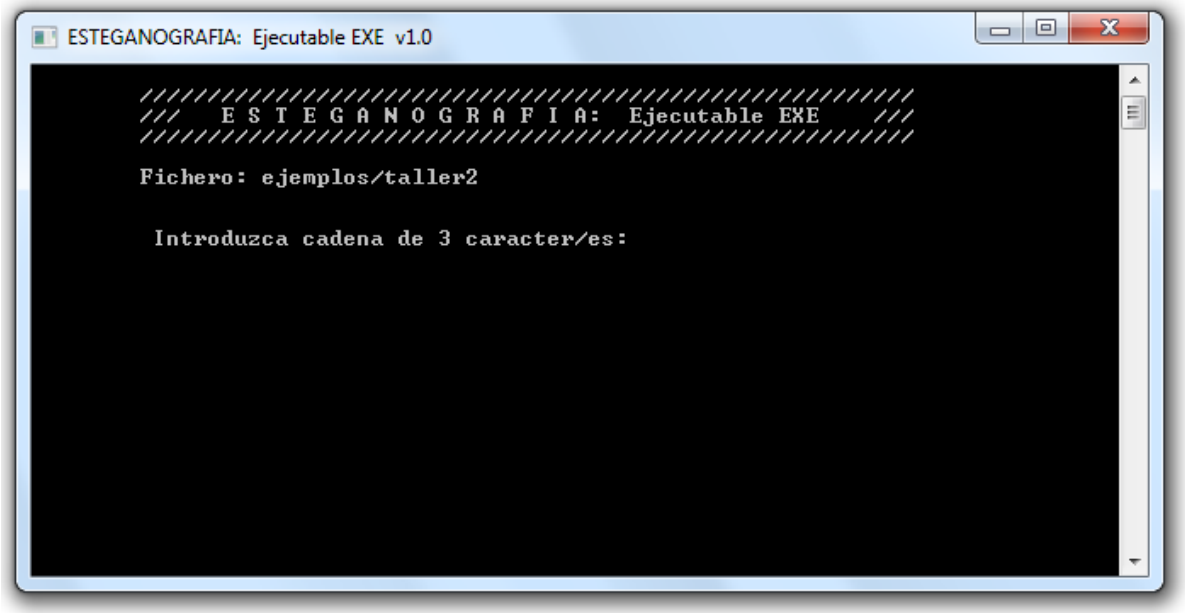


Ilustración 89: Pantalla de solicitud de cadena de caracteres para ejecutable EXE

LONGITUD EXACTA:

El programa solicita una cadena de 3 caracteres, se introduce la cadena "irc" con una longitud exacta de 3 caracteres, el programa la lee y la utiliza sin problemas.

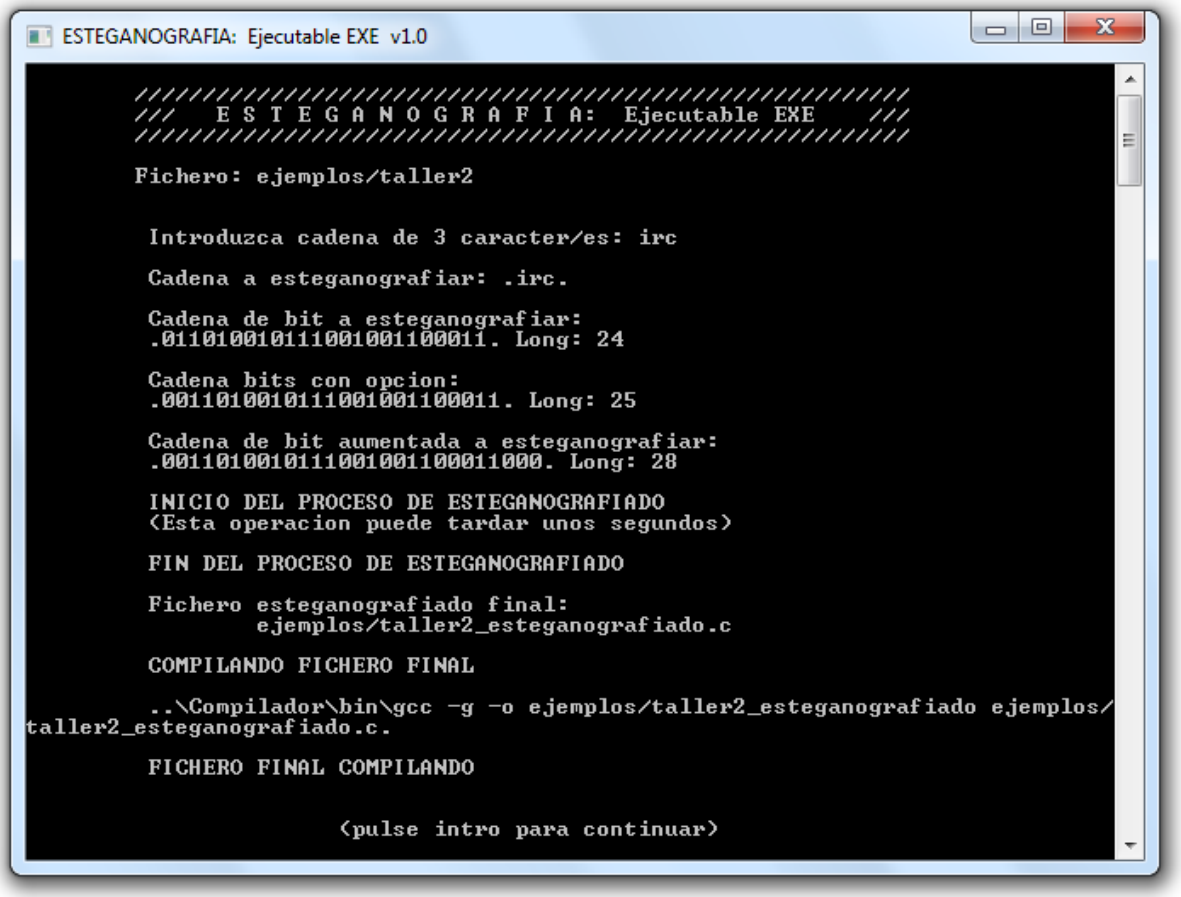


Ilustración 90: Prueba de esteganografiado longitud exacta para caracteres en ejecutable EXE

Para verificar que el programa ha utilizado la cadena introducida sin problemas desesteganografiamos para obtener de nuevo la cadena y observamos que esta sigue siendo la cadena "irc" con una longitud exacta de 3 caracteres.

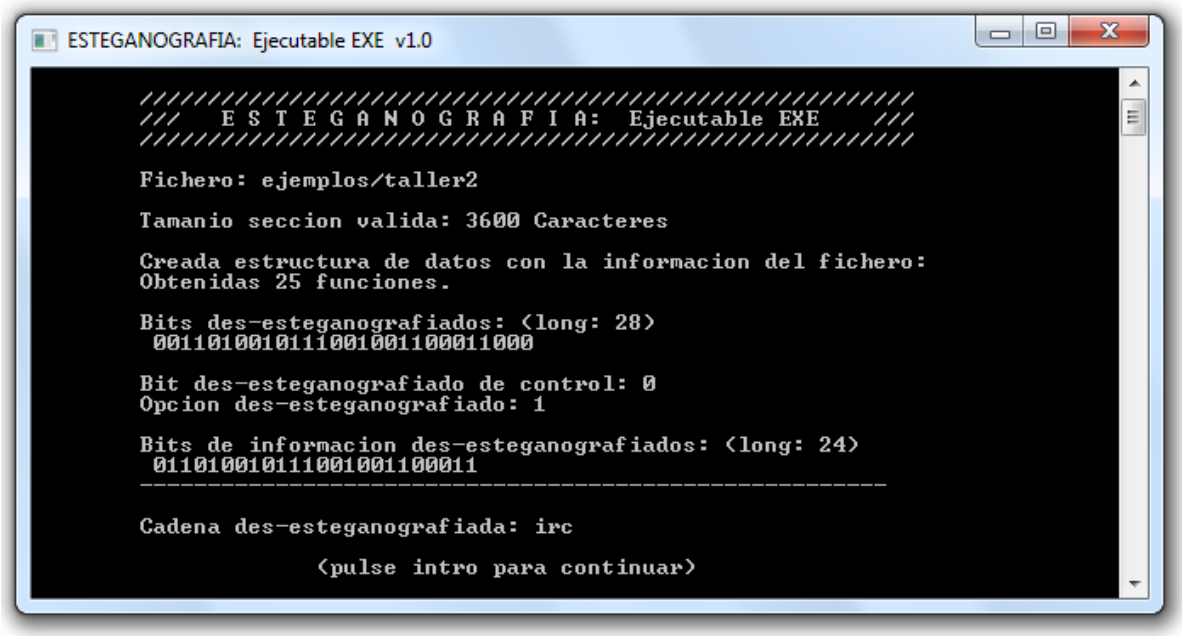


Ilustración 91: Prueba de desesteganografiado longitud exacta para caracteres en ejecutable EXE

LONGITUD INFERIOR:

El programa solicita una cadena de 3 caracteres, se introduce la cadena "IR" con una longitud inferior, en este caso de solo 2 caracteres, el programa la lee, detecta que su longitud es inferior, la aumenta concatenándola 1 espacios en blanco y la utiliza sin problemas.

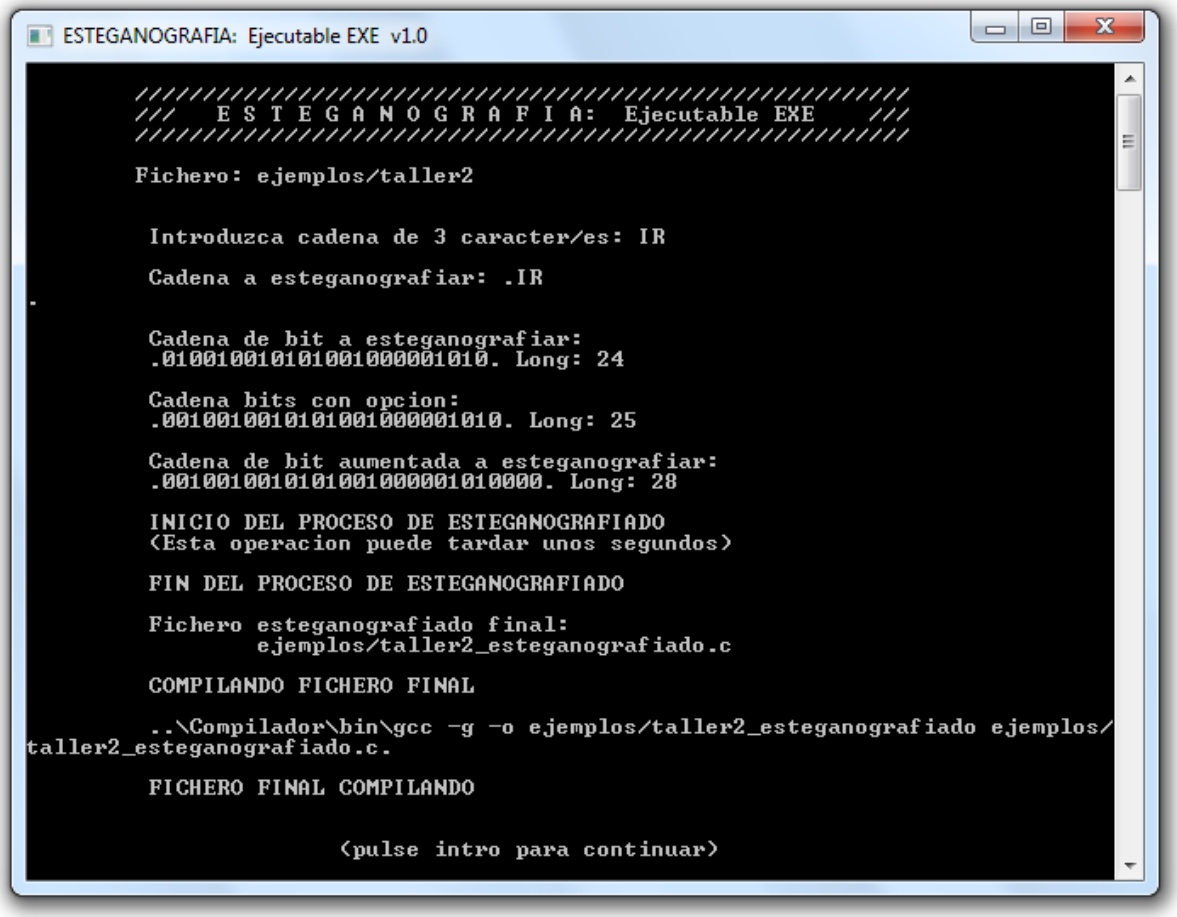


Ilustración 92: Prueba de esteganografiado longitud inferior para caracteres en ejecutable EXE

LONGITUD SUPERIOR:

El programa solicita una cadena de 3 caracteres, se introduce la cadena “Ivan” con una longitud superior, en este caso 4 caracteres, el programa la lee, detecta que su longitud es superior, la trunca a la longitud requerida de 3 caracteres y la utiliza sin problemas.

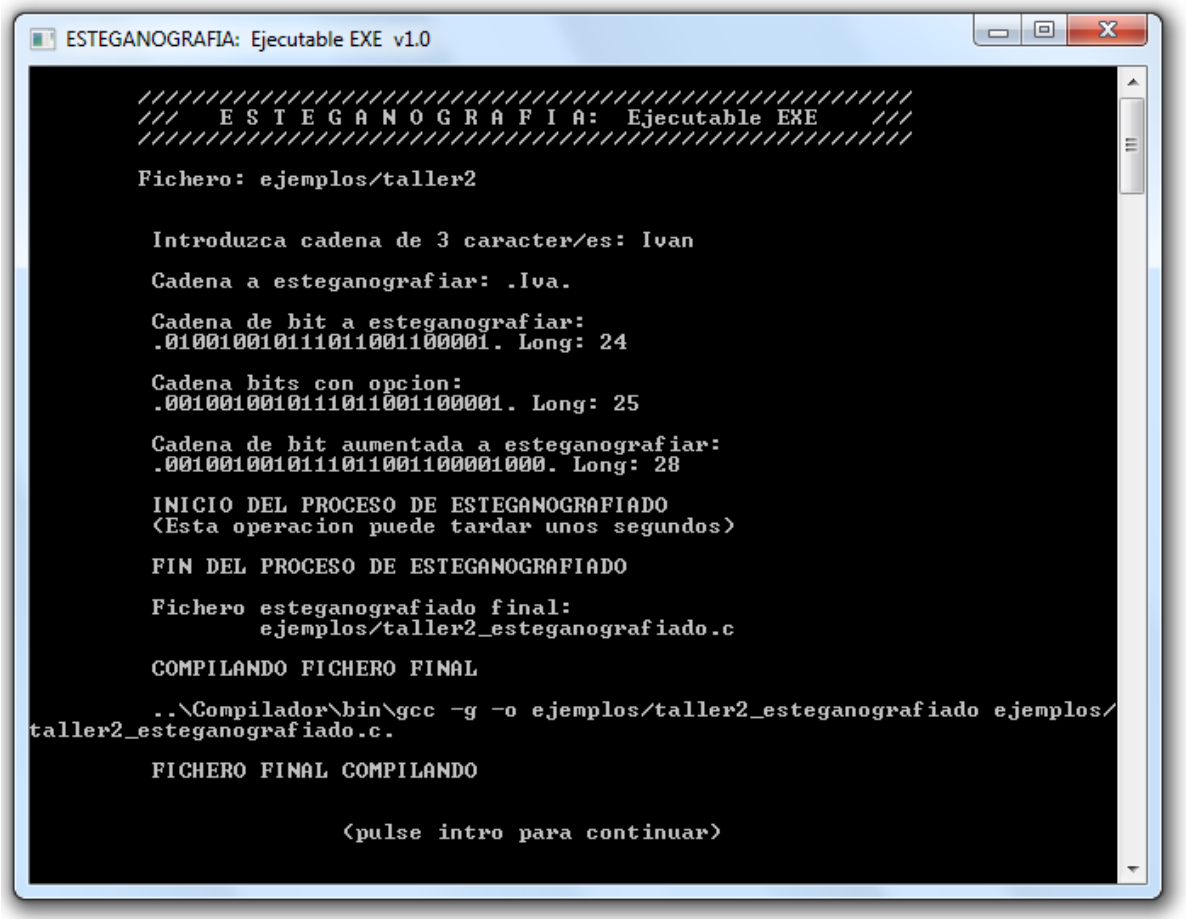


Ilustración 93: Prueba de esteganografiado longitud superior para caracteres en ejecutable EXE

LONGITUD CERO:

El programa solicita una cadena de 3 caracteres y no se introduce cadena alguna, el programa la lee como una cadena vacía y actúa como si de una cadena de longitud inferior se tratara, la aumenta de cero a la longitud requerida concatenándola 3 espacios en blanco y la utiliza sin problemas.

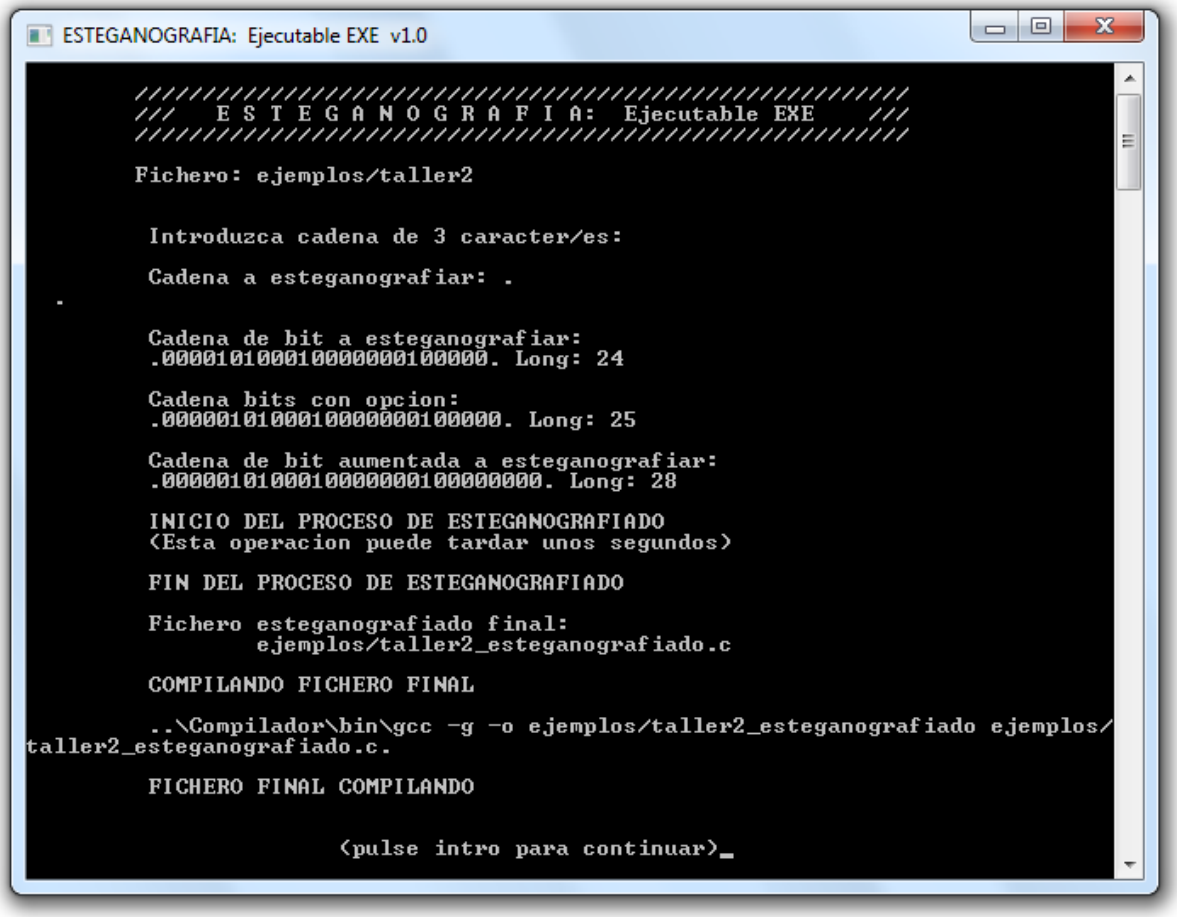


Ilustración 94: Prueba de esteganografiado longitud cero para caracteres en ejecutable EXE

IDENTIFICADOR	PRUEBA 5	
NOMBRE	SOLICITUD CADENA MAYÚSCULAS PARA EJECUTABLE EXE	
PROPÓSITO	El programa ha de ser flexible a la hora de solicitar la cadena de mayúsculas a esteganografiar. El programa no ha de fallar o funcionar mal ante cualquier tipo de longitud de entrada. Siendo a elección del usuario el utilizar toda la capacidad disponible o no.	
CASOS DE PRUEBA	SALIDA	RESULTADO
Longitud exacta	El programa utiliza para el esteganografiado la cadena de mayúsculas introducida.	Correcto
Longitud inferior	El programa utiliza para el esteganografiado la cadena de mayúsculas introducida y rellena con espacios en blanco hasta la longitud pedida.	Correcto
Longitud superior	El programa utiliza para el esteganografiado la cadena de mayúsculas introducida truncándola a la longitud pedida.	Correcto
Longitud cero	El programa actúa como si la cadena fuera de longitud menor y rellena con espacios en blanco hasta la longitud pedida en este caso la longitud total.	Correcto

Tabla 26: Prueba solicitud cadena mayúsculas para ejecutable EXE

PANTALLA DE SOLICITUD DE CADENA PARA EL ESTEGANOGRAFIADO DE MAYÚSCULAS EN EJECUTABLE EXE:

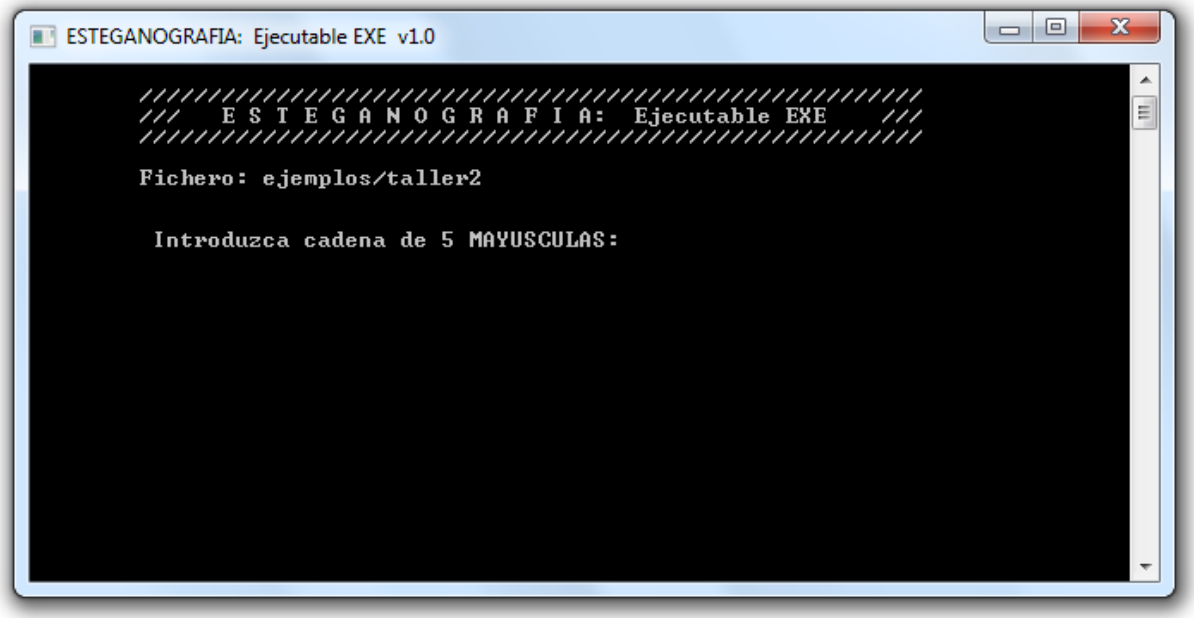


Ilustración 95: Pantalla de solicitud de cadena de mayúsculas para ejecutable EXE

LONGITUD EXACTA:

El programa solicita una cadena de 5 mayúsculas, se introduce la cadena "IVANR" con una longitud exacta de 5 caracteres, el programa la lee y la utiliza sin problemas.

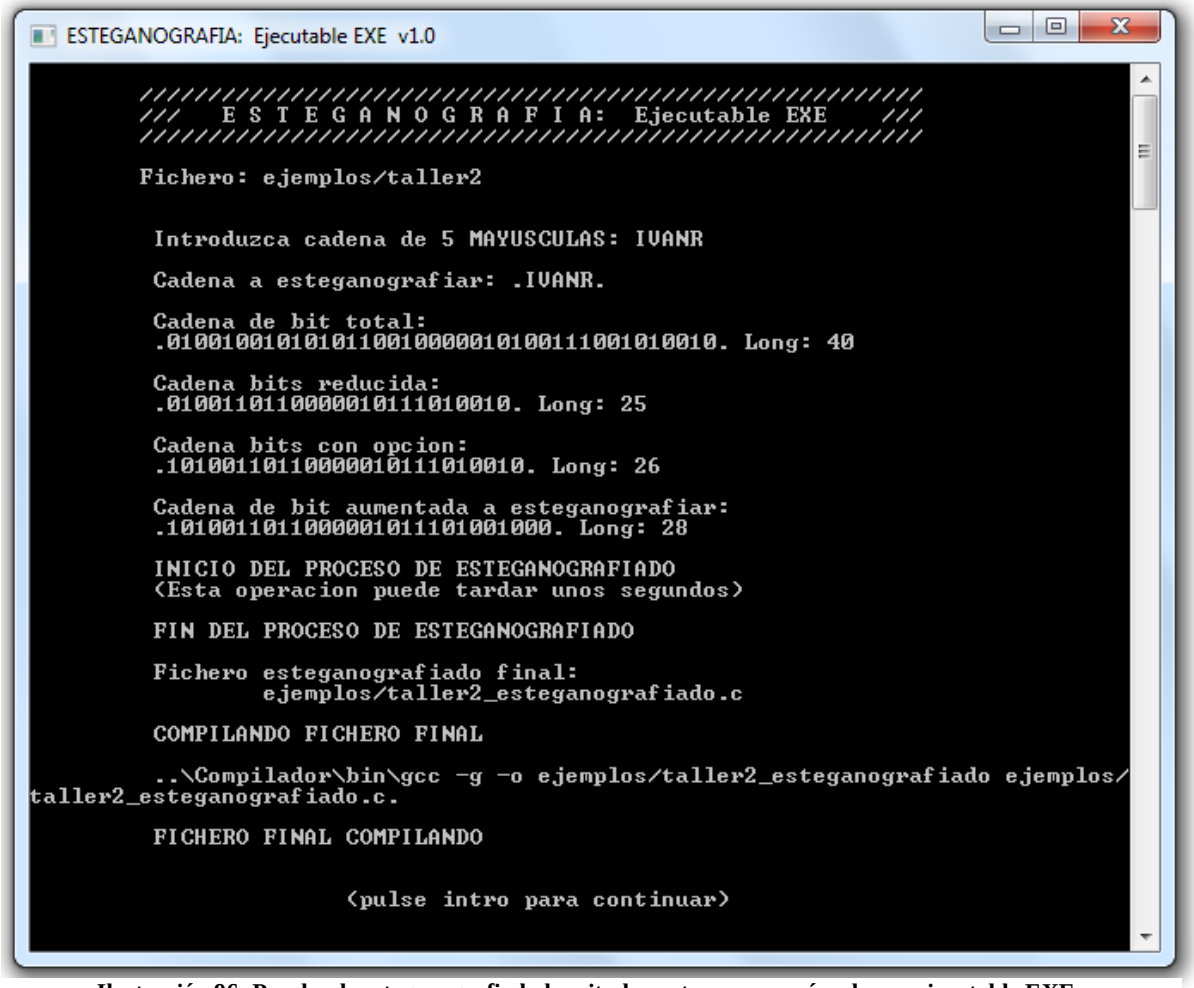


Ilustración 96: Prueba de esteganografiado longitud exacta para mayúsculas en ejecutable EXE

Para verificar que el programa ha utilizado la cadena introducida sin problemas desesteganografiamos para obtener de nuevo la cadena y observamos que esta sigue siendo la cadena "IVANR" con una longitud exacta de 5 caracteres.

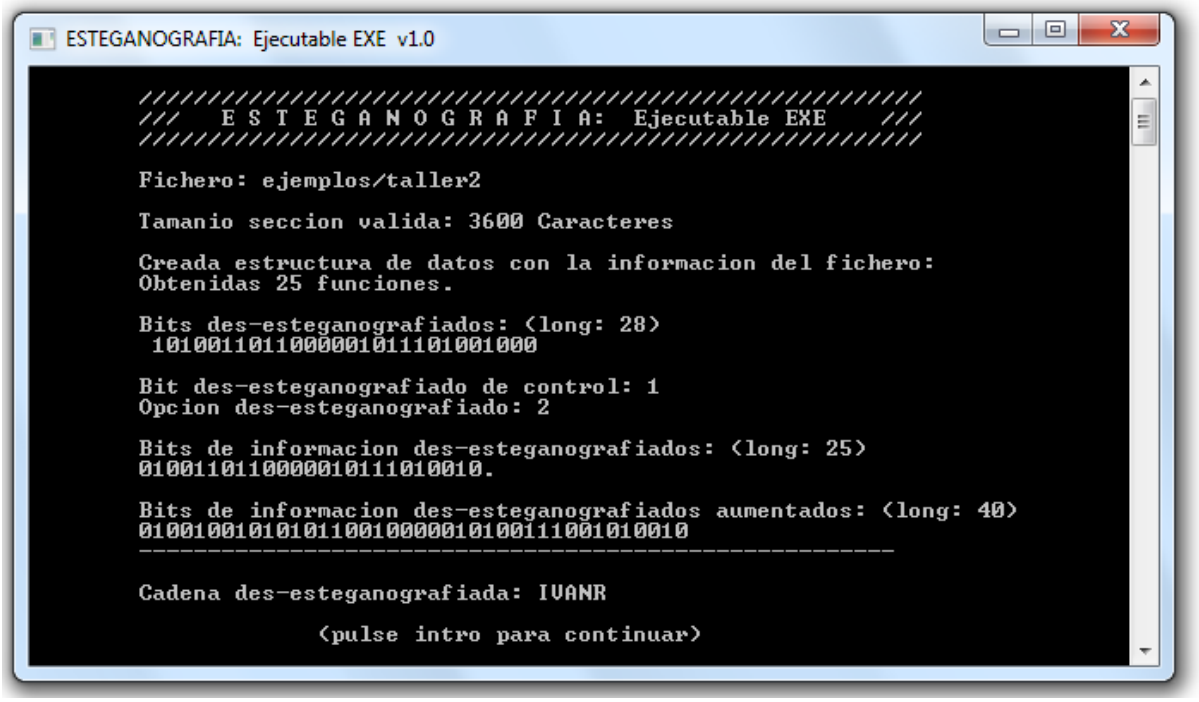


Ilustración 97: Prueba de desesteganografiado longitud exacta para mayúsculas en ejecutable EXE

LONGITUD INFERIOR:

El programa solicita una cadena de 5 mayúsculas, se introduce la cadena "PFC" con una longitud inferior, en este caso de solo 3 caracteres, el programa la lee, detecta que su longitud es inferior, la aumenta concatenándola 2 espacios en blanco y la utiliza sin problemas.

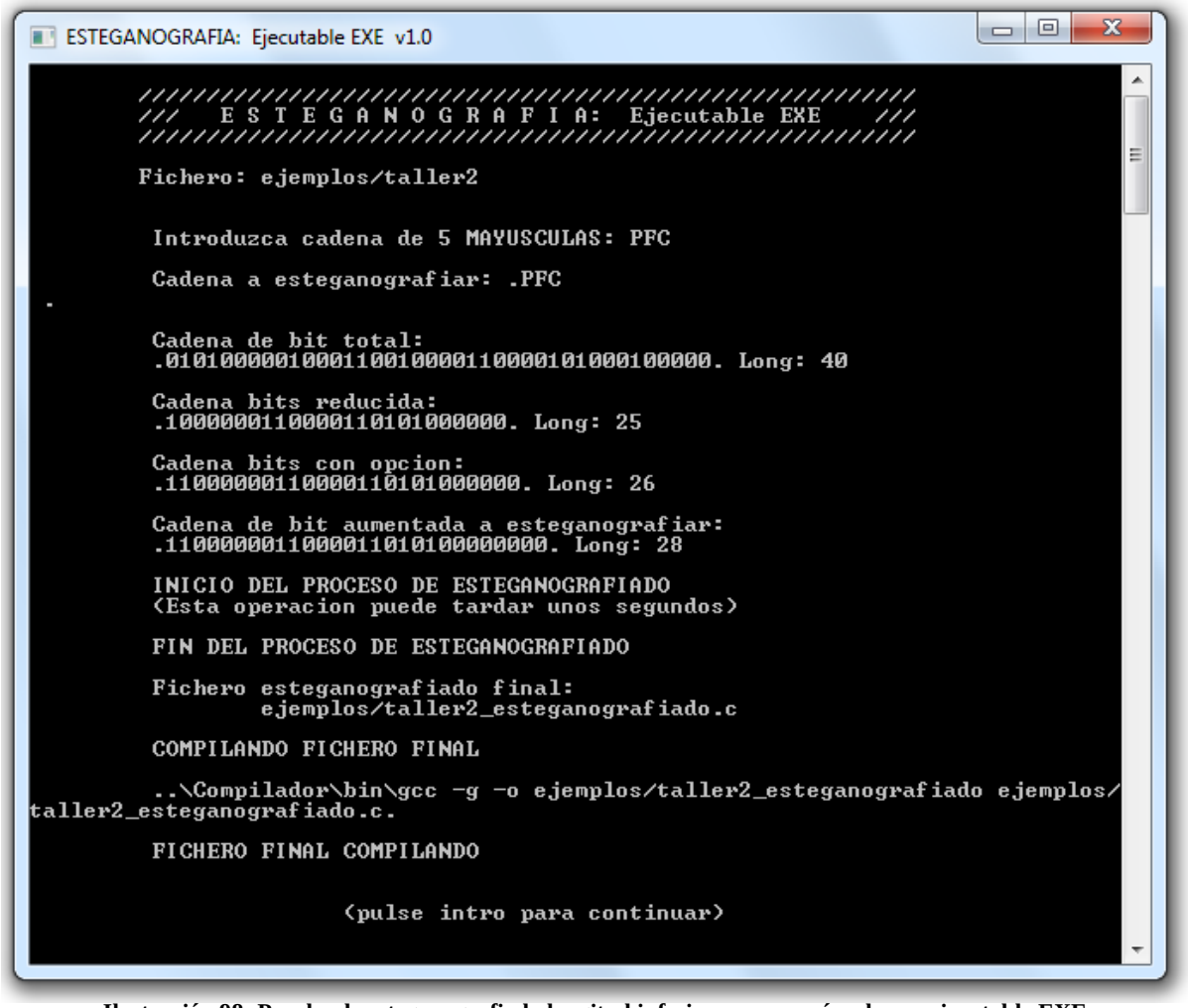


Ilustración 98: Prueba de esteganografiado longitud inferior para mayúsculas en ejecutable EXE

LONGITUD SUPERIOR:

El programa solicita una cadena de 5 mayúsculas, se introduce la cadena “PFCIVAN” con una longitud superior, en este caso 7 caracteres, el programa la lee, detecta que su longitud es superior, la trunca a la longitud requerida de 5 mayúsculas y la utiliza sin problemas.

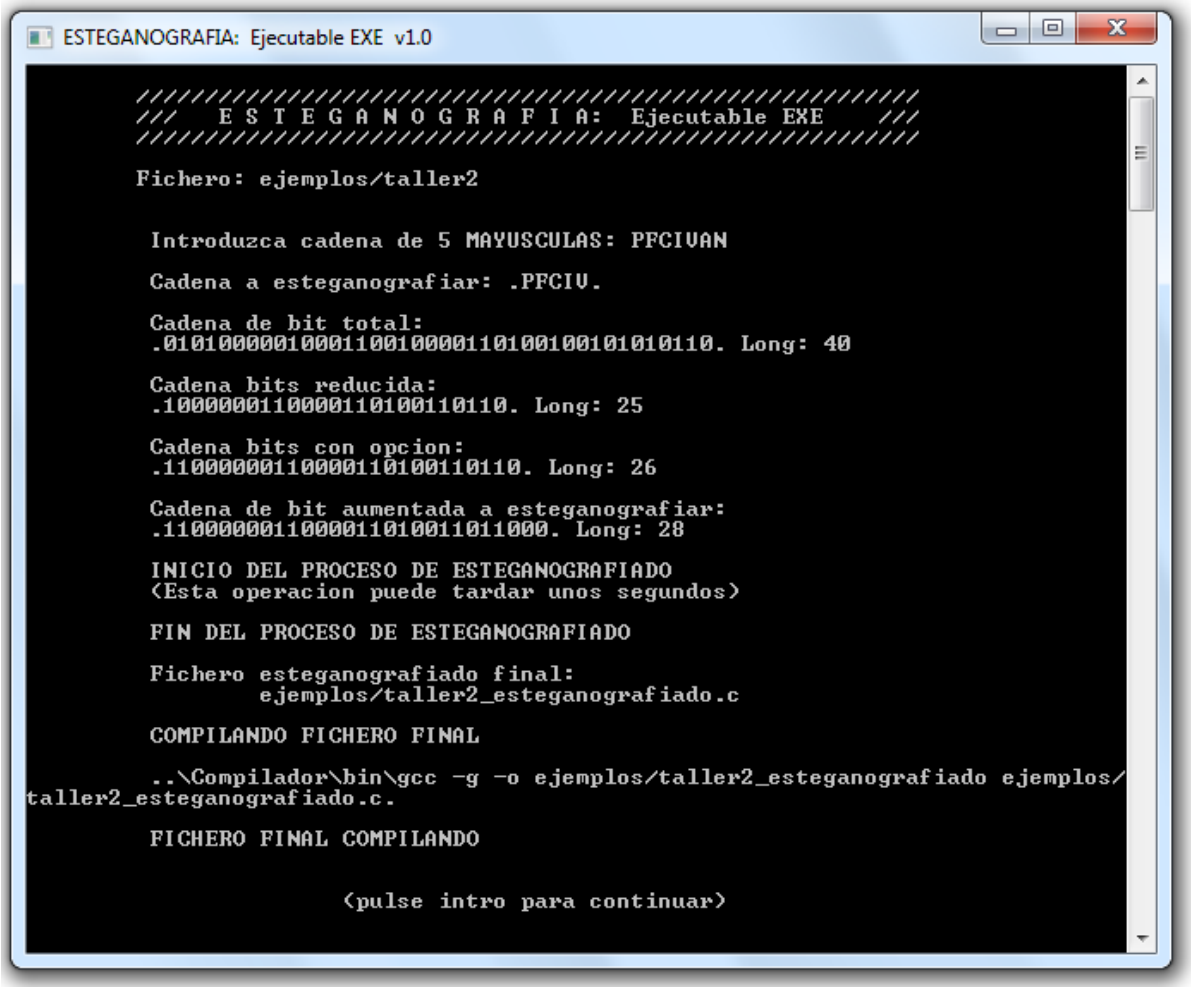


Ilustración 99: Prueba de esteganografiado longitud superior para mayúsculas en ejecutable EXE

LONGITUD CERO:

El programa solicita una cadena de 5 mayúsculas y no se introduce cadena alguna, el programa la lee como una cadena vacía y actúa como si de una cadena de longitud inferior se tratara, la aumenta de cero a la longitud requerida concatenándola 5 espacios en blanco y la utiliza sin problemas.

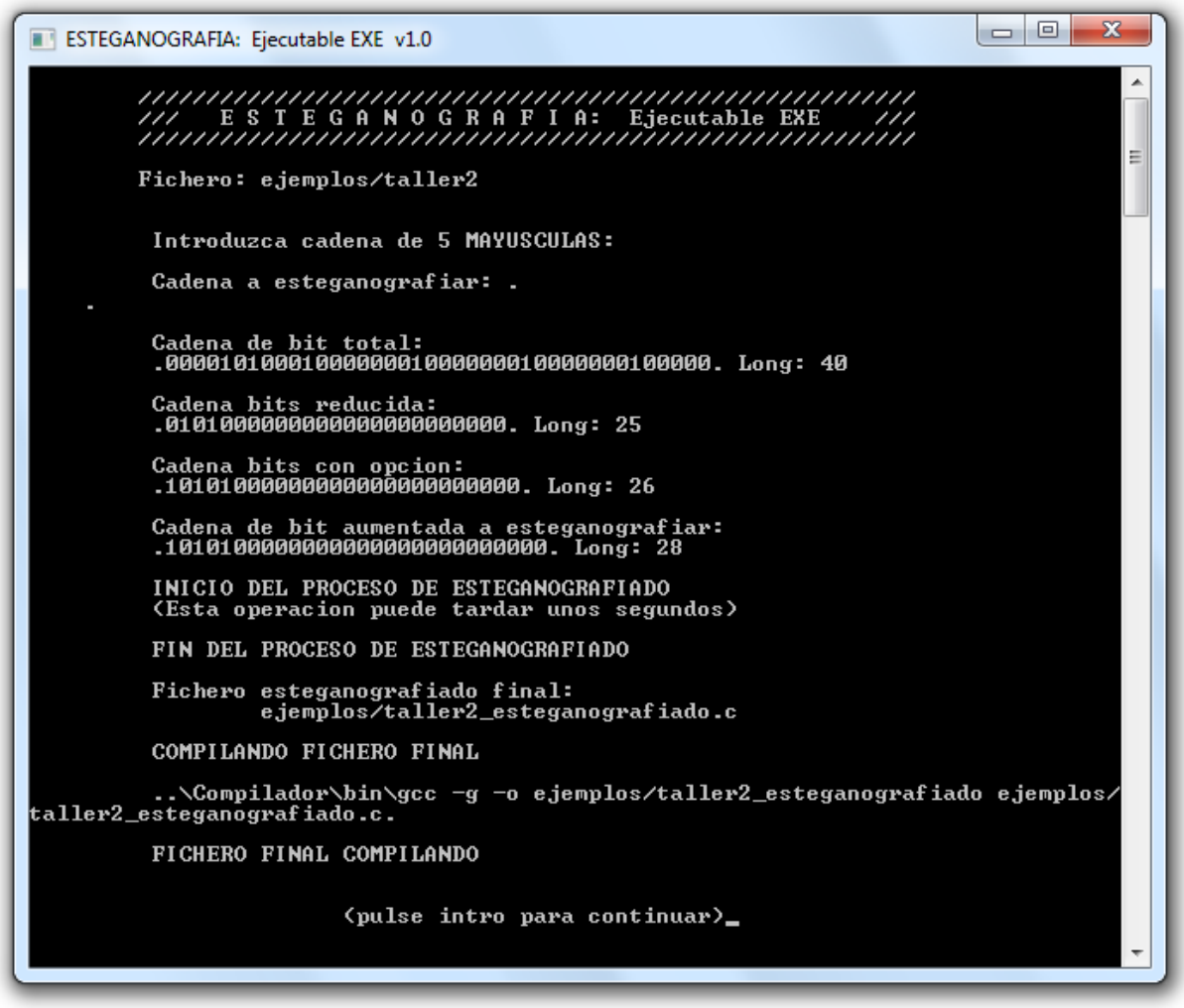


Ilustración 100: Prueba de esteganografiado longitud cero para mayúsculas en ejecutable EXE

IDENTIFICADOR	PRUEBA 6	
NOMBRE	SOLICITUD CADENA CARACTERES PARA CÓDIGO C AMPLIADO	
PROPÓSITO	El programa ha de ser flexible a la hora de solicitar la cadena de caracteres a esteganografiar. El programa no ha de fallar o funcionar mal ante cualquier tipo de longitud de entrada. Siendo a elección del usuario el utilizar toda la capacidad disponible o no.	
CASOS DE PRUEBA	SALIDA	RESULTADO
Longitud exacta	El programa utiliza para el esteganografiado la cadena introducida.	Correcto
Longitud inferior	El programa utiliza para el esteganografiado la cadena introducida y rellena con espacios en blanco hasta la longitud pedida.	Correcto
Longitud superior	El programa utiliza para el esteganografiado la cadena introducida truncándola a la longitud pedida.	Correcto
Longitud cero	El programa actúa como si la cadena fuera de longitud menor y rellena con espacios en blanco hasta la longitud pedida en este caso la longitud total.	Correcto

Tabla 27: Prueba solicitud cadena caracteres para código c ampliado

PANTALLA DE SOLICITUD DE CADENA PARA EL ESTEGANOGRAFIADO DE CARACTERES EN CÓDIGO C AMPLIADO:

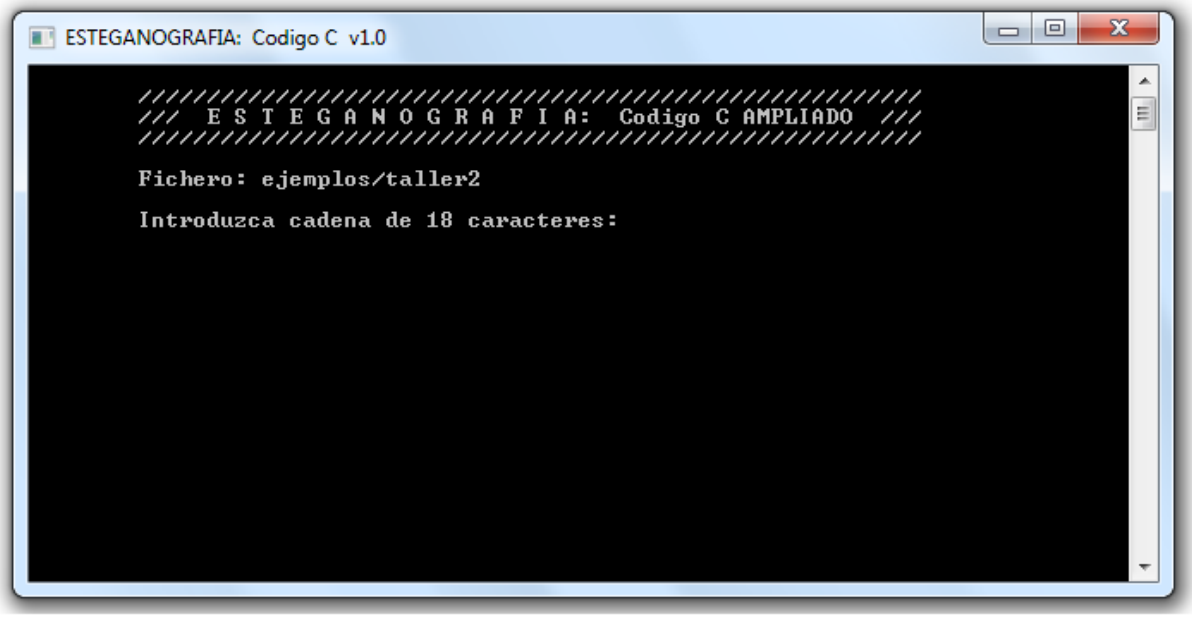


Ilustración 101: Pantalla de solicitud de cadena de caracteres para código C ampliado

LONGITUD EXACTA:

El programa solicita una cadena de 18 caracteres, se introduce la cadena "PFCv1 Ivan Redondo" con una longitud exacta de 18 caracteres, el programa la lee y la utiliza sin problemas.

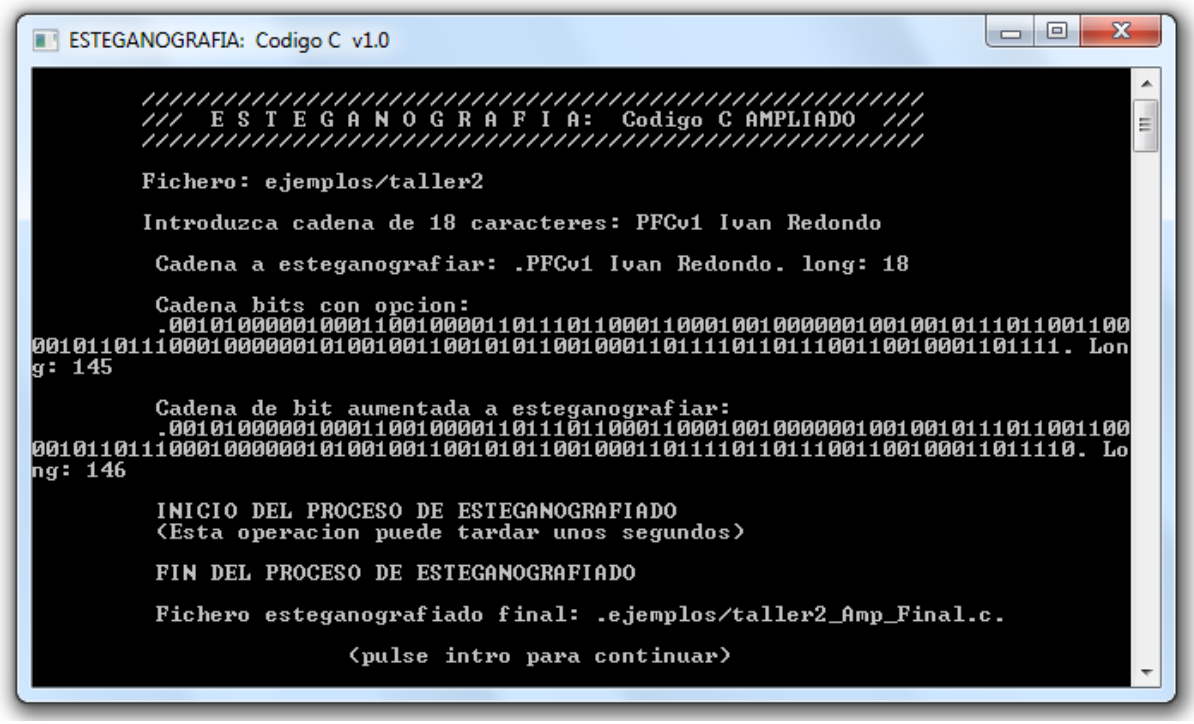


Ilustración 102: Prueba de esteganografiado longitud exacta para caracteres en código C ampliado

Para verificar que el programa ha utilizado la cadena introducida sin problemas desesteganografiamos para obtener de nuevo la cadena y observamos que esta sigue siendo la cadena "PFCv1 Ivan Redondo" con una longitud exacta de 18 caracteres.

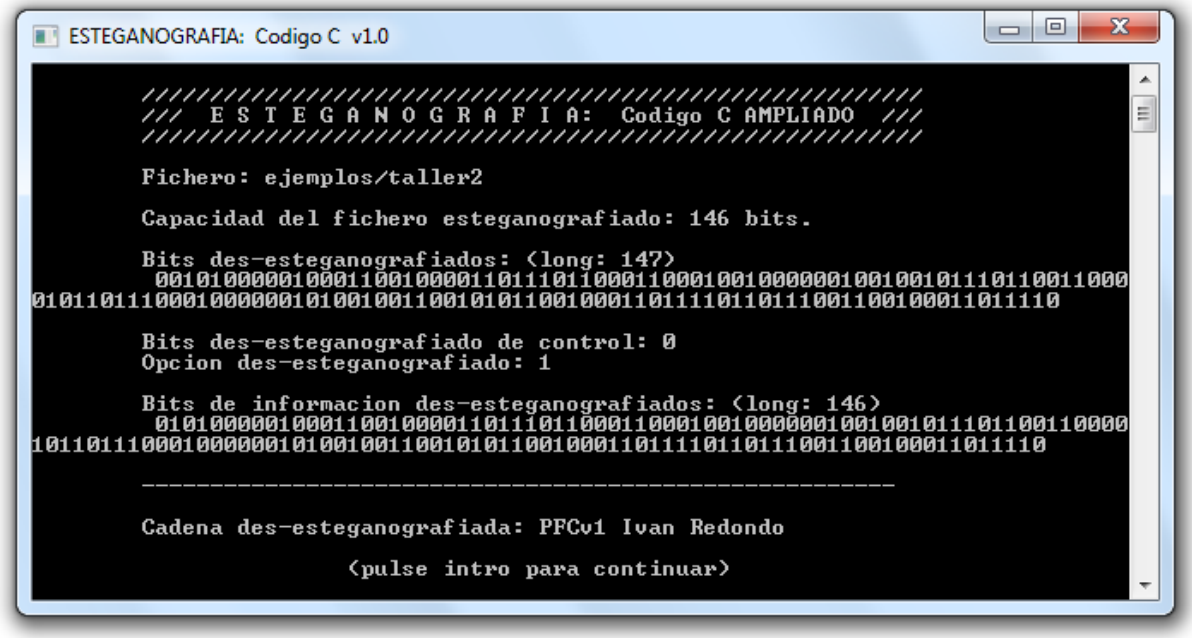


Ilustración 103: Prueba de desesteganografiado longitud exacta para caracteres en código C ampliado

LONGITUD INFERIOR:

El programa solicita una cadena de 14 caracteres, se introduce la cadena “Ivan Redondo” con una longitud inferior, en este caso de solo 7 caracteres, el programa la lee, detecta que su longitud es inferior, la aumenta concatenándola 7 espacios en blanco y la utiliza sin problemas.

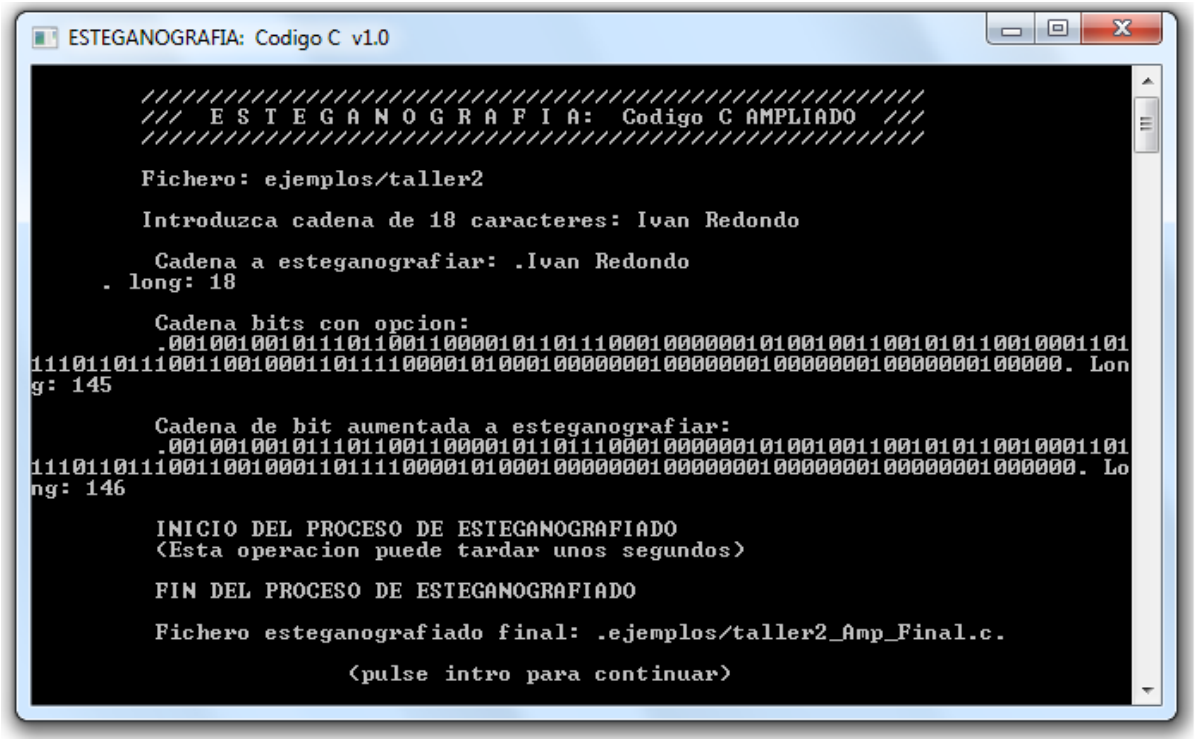


Ilustración 104: Prueba de esteganografiado longitud inferior para caracteres en código C ampliado

LONGITUD SUPERIOR:

El programa solicita una cadena de 14 caracteres, se introduce la cadena “PFC Ivan Redondo Chisvert” con una longitud superior, en este caso 21 caracteres, el programa la lee, detecta que su longitud es superior, la trunca a la longitud requerida de 14 caracteres y la utiliza sin problemas.

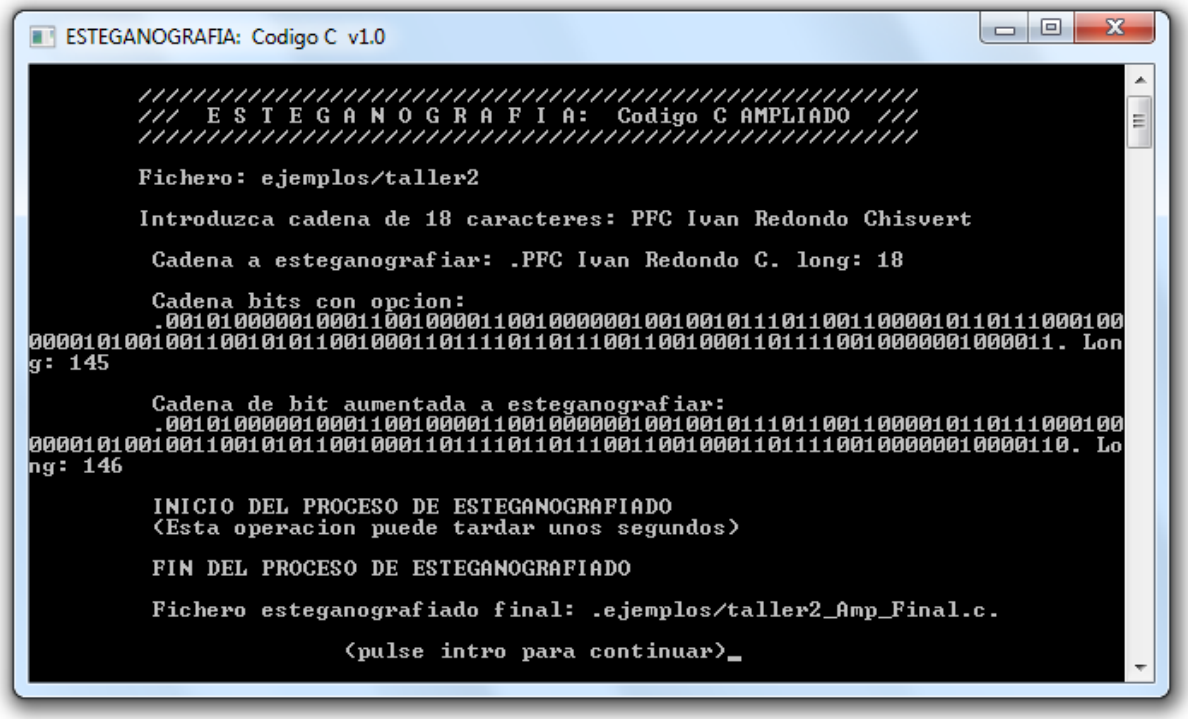
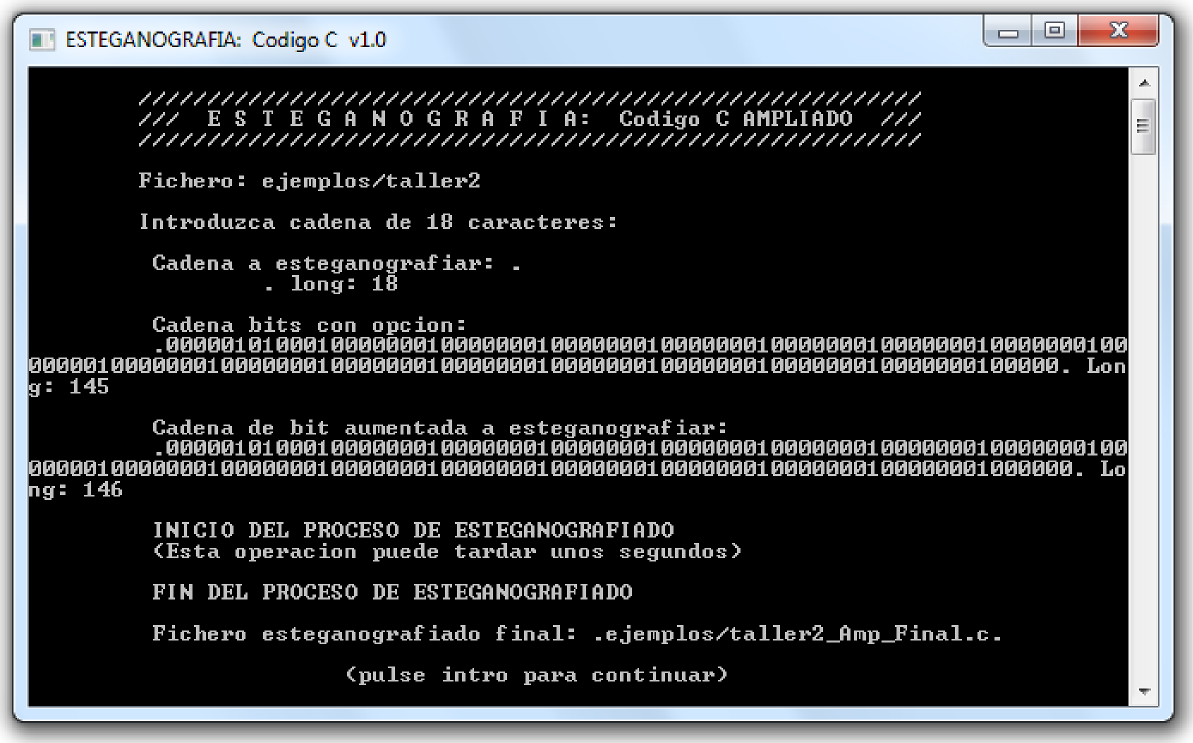


Ilustración 105: Prueba de esteganografiado longitud superior para caracteres en código C ampliado

LONGITUD CERO:

El programa solicita una cadena de 14 caracteres y no se introduce cadena alguna, el programa la lee como una cadena vacía y actúa como si de una cadena de longitud inferior se tratara, la aumenta de cero a la longitud requerida concatenándola 14 espacios en blanco y la utiliza sin problemas.


```
ESTEGANOGRAFIA: Codigo C v1.0

        ////////////////////////////////////////////////////////////////
        ///  E S T E G A N O G R A F I A:  Codigo C AMPLIADO  ///
        ////////////////////////////////////////////////////////////////

        Fichero: ejemplos/taller2

        Introduzca cadena de 18 caracteres:

         Cadena a esteganografiar: .
                 . long: 18

         Cadena bits con opcion:
         .0000010100010000000100000001000000010000000100000001000000010000000100
00000100000001000000010000000100000001000000010000000100000001000000010000000. Lon
g: 145

         Cadena de bit aumentada a esteganografiar:
         .0000010100010000000100000001000000010000000100000001000000010000000100
000001000000010000000100000001000000010000000100000001000000010000000100000000. Lo
ng: 146

         INICIO DEL PROCESO DE ESTEGANOGRAFIADO
         (Esta operacion puede tardar unos segundos)

         FIN DEL PROCESO DE ESTEGANOGRAFIADO

         Fichero esteganografiado final: .ejemplos/taller2_Amp_Final.c.

                       (pulse intro para continuar)
```

Ilustración 106: Prueba de esteganografiado longitud cero para caracteres en código C ampliado

IDENTIFICADOR	PRUEBA 7	
NOMBRE	SOLICITUD CADENA MAYÚSCULAS PARA CÓDIGO C AMPLIADO	
PROPÓSITO	El programa ha de ser flexible a la hora de solicitar la cadena de mayúsculas a esteganografiar. El programa no ha de fallar o funcionar mal ante cualquier tipo de longitud de entrada. Siendo a elección del usuario el utilizar toda la capacidad disponible o no.	
CASOS DE PRUEBA	SALIDA	RESULTADO
Longitud exacta	El programa utiliza para el esteganografiado la cadena de mayúsculas introducida.	Correcto
Longitud inferior	El programa utiliza para el esteganografiado la cadena de mayúsculas introducida y rellena con espacios en blanco hasta la longitud pedida.	Correcto
Longitud superior	El programa utiliza para el esteganografiado la cadena de mayúsculas introducida truncándola a la longitud pedida.	Correcto
Longitud cero	El programa actúa como si la cadena fuera de longitud menor y rellena con espacios en blanco hasta la longitud pedida en este caso la longitud total.	Correcto

Tabla 28: Prueba solicitud cadena mayúsculas para código c ampliado

PANTALLA DE SOLICITUD DE CADENA PARA EL ESTEGANOGRAFIADO DE MAYÚSCULAS EN CÓDIGO C AMPLIADO:

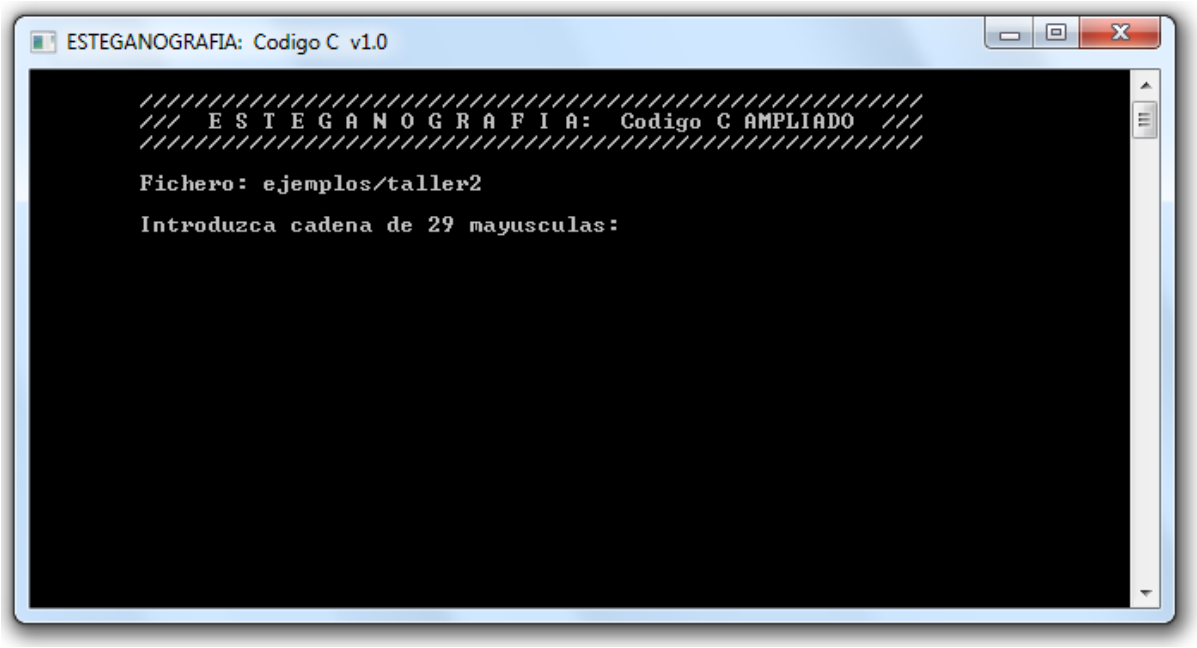


Ilustración 107: Pantalla de solicitud de cadena de mayúsculas para código C ampliado

LONGITUD EXACTA:

El programa solicita una cadena de 29 mayúsculas, se introduce la cadena “VERSION IVAN REDONDO CHISVERT” con una longitud exacta de 29 caracteres, el programa la lee y la utiliza sin problemas.

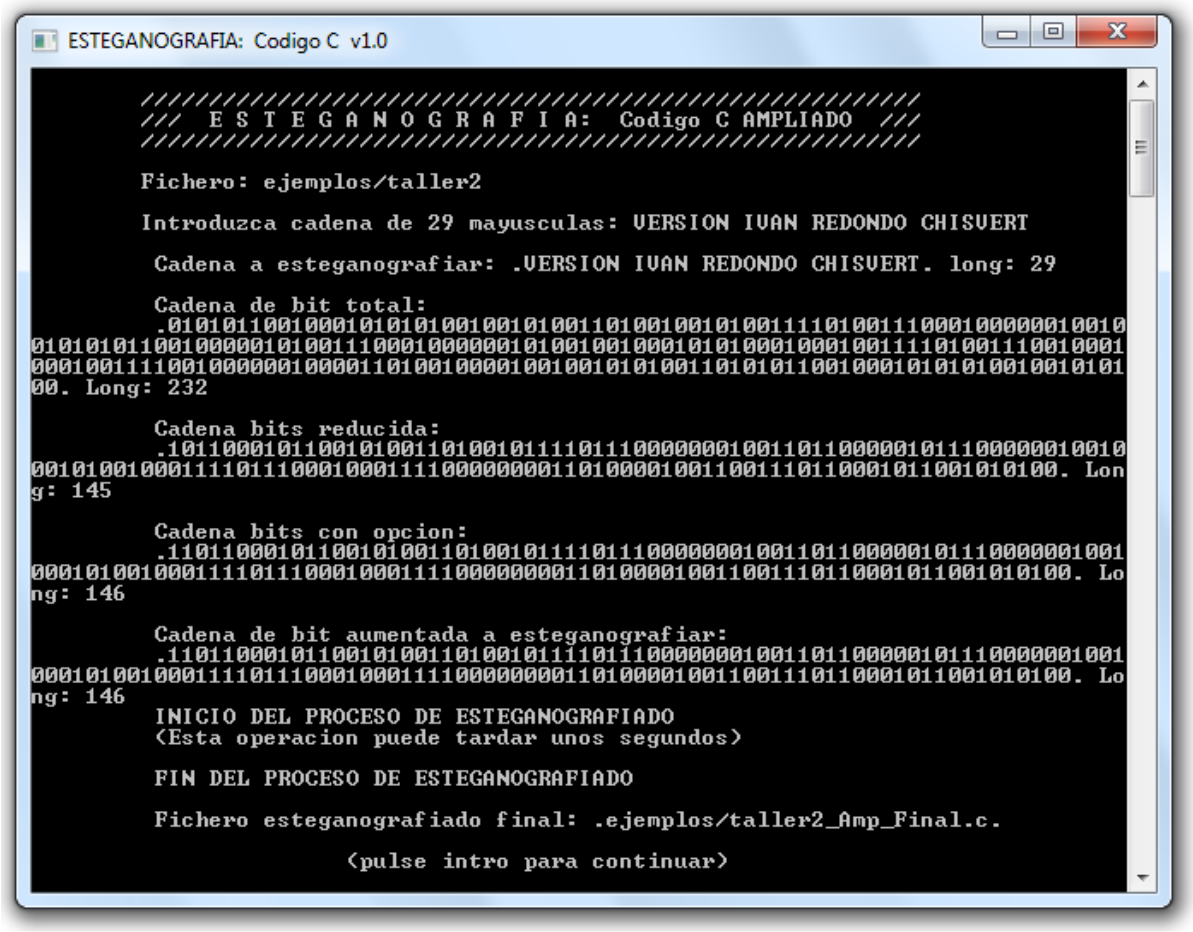


Ilustración 108: Prueba de esteganografiado longitud exacta para mayúsculas en código C ampliado

Para verificar que el programa ha utilizado la cadena introducida sin problemas desesteganografiamos para obtener de nuevo la cadena y observamos que la cadena resultante es la siguiente, "VERSION@IVAN@REDONDO@CHISVERT", esta cadena es similar a la cadena anterior con la diferencia de que los espacios en blanco los desesteganografía en la versión de mayúsculas como "@" arrobas debido al algoritmo de codificación.

Esto no se califica como un error puesto que lo que se pide son mayúsculas y el espacio en blanco no se considera como tal.

Se podría haber eliminado esto del programa final simplemente sustituyendo en la cadena desesteganografiada las arrobas por espacios en blanco, pero no se consideró oportuno el falseo o sustitución forzada de caracteres en el desesteganografiado puesto que esto podría afectar a la veracidad del firmado o de su demostración puesto que no coincide lo esteganografiado con lo supuestamente desesteganografiado.

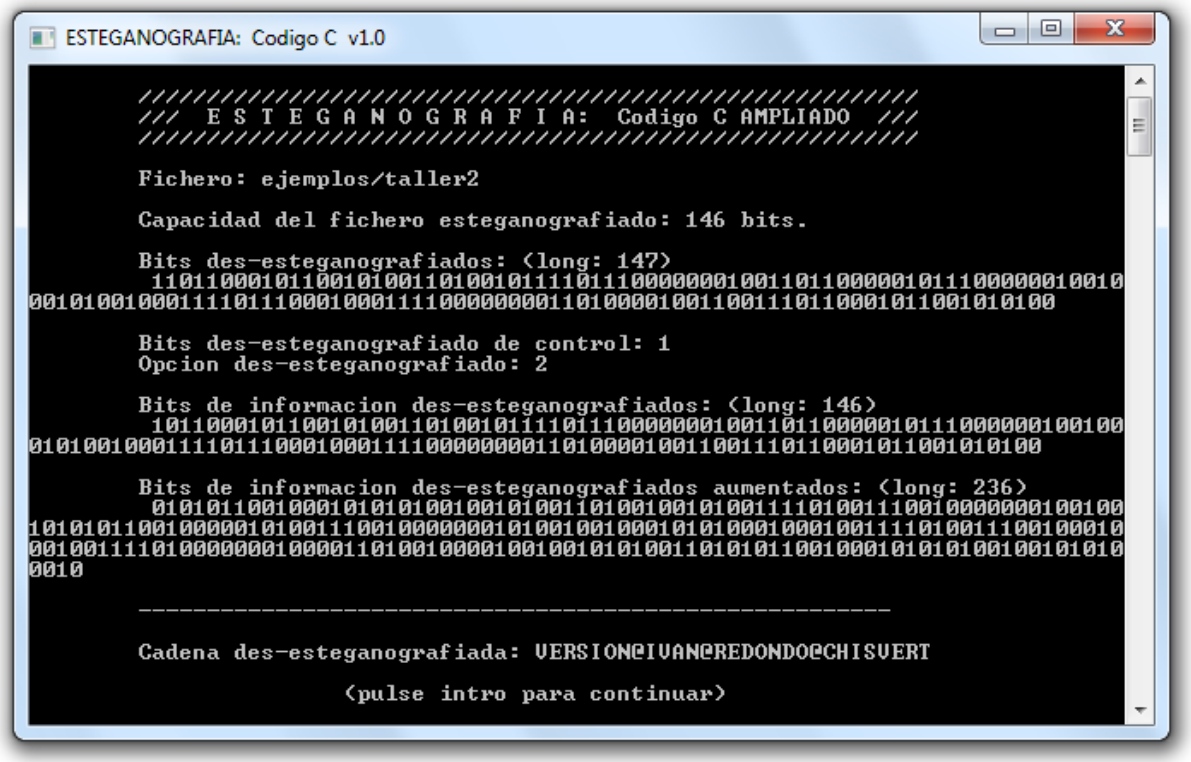


Ilustración 109: Prueba de desesteganografiado longitud exacta para mayúsculas en código C ampliado

LONGITUD INFERIOR:

El programa solicita una cadena de 29 mayúsculas, se introduce la cadena “IVAN REDONDO CHISVERT” con una longitud inferior, en este caso de solo 21 caracteres, el programa la lee, detecta que su longitud es inferior, la aumenta concatenándola 8 espacios en blanco y la utiliza sin problemas.

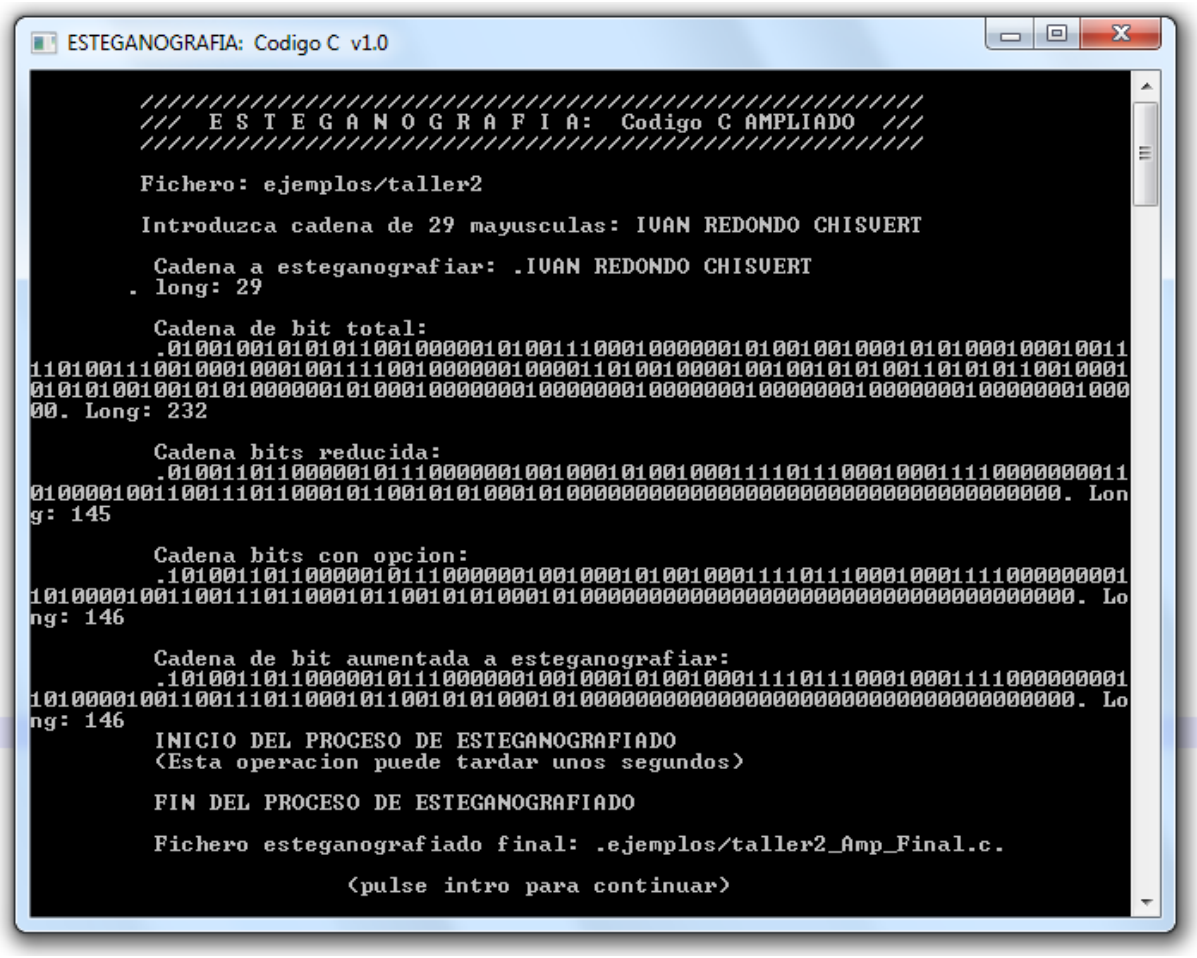


Ilustración 110: Prueba de esteganografiado longitud inferior para mayúsculas en código C ampliado

LONGITUD SUPERIOR:

El programa solicita una cadena de 29 mayúsculas, se introduce la cadena “PROPIEDAD DE IVAN REDONDO CHISVERT” con una longitud superior, en este caso 34 caracteres, el programa la lee, detecta que su longitud es superior, la trunca a la longitud requerida de 29 mayúsculas y la utiliza sin problemas.

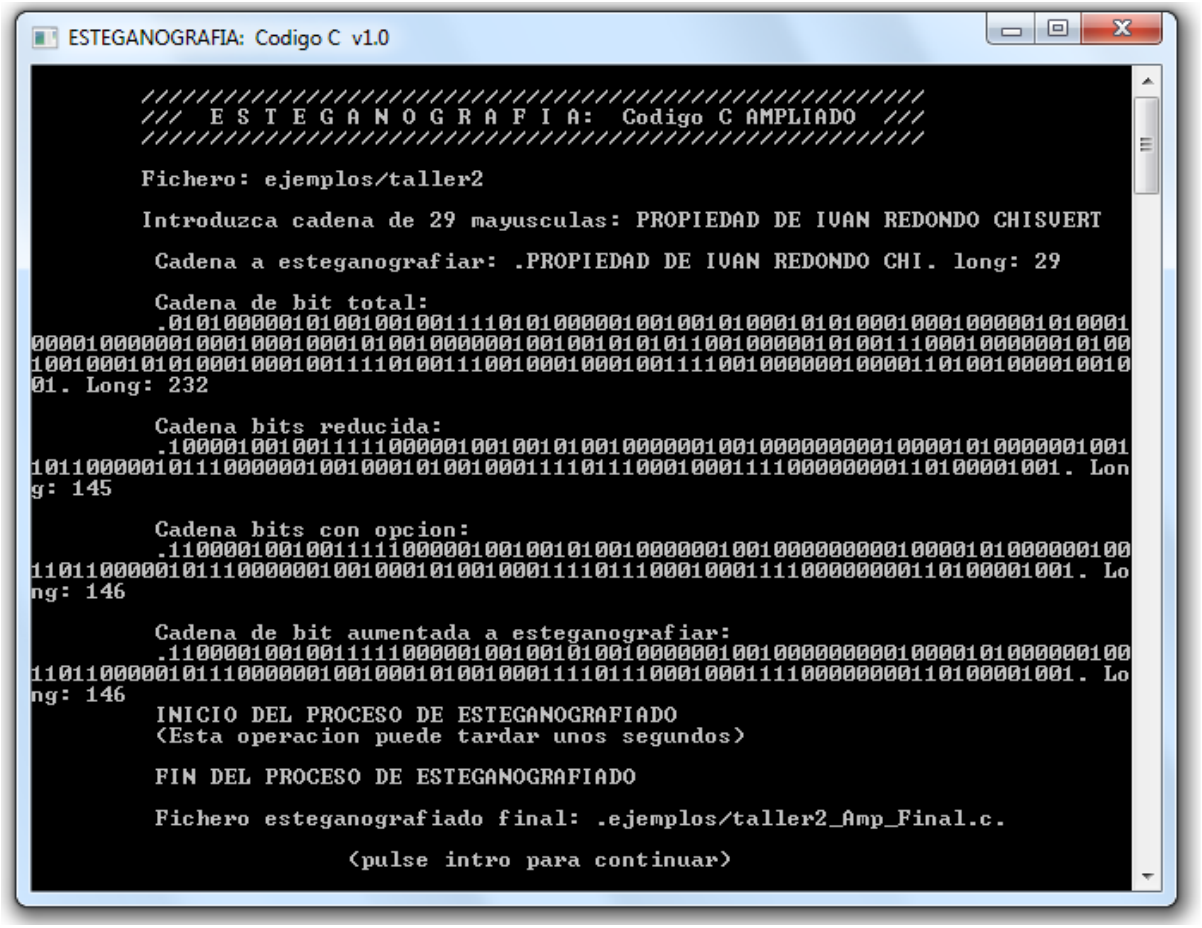
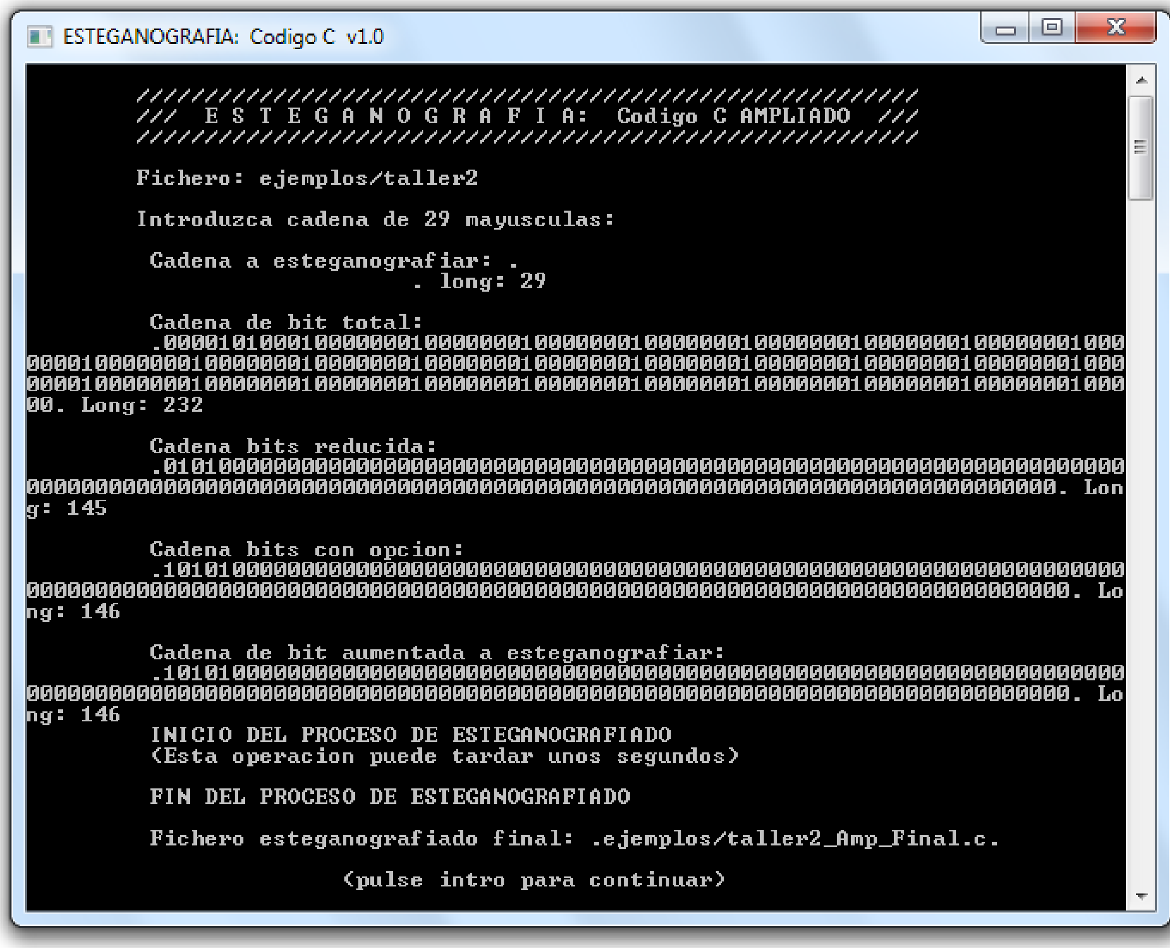


Ilustración 111: Prueba de esteganografiado longitud superior para mayúsculas en código C ampliado

LONGITUD CERO:

El programa solicita una cadena de 23 mayúsculas y no se introduce cadena alguna, el programa la lee como una cadena vacía y actúa como si de una cadena de longitud inferior se tratara, la aumenta de cero a la longitud requerida concatenándola 23 espacios en blanco y la utiliza sin problemas.



```
ESTEGANOGRAFIA:  Codigo C  v1.0

        //////////////////////////////////////////////////////////
        ///  E S T E G A N O G R A F I A:  Codigo C AMPLIADO  ///
        //////////////////////////////////////////////////////////

        Fichero: ejemplos/taller2

        Introduzca cadena de 29 mayusculas:

         Cadena a esteganografiar: .
                            . long: 29

         Cadena de bit total:
         .0000101000100000001000000010000000100000001000000010000000100000001000
00001000000010000000100000001000000010000000100000001000000010000000100000001000
00001000000010000000100000001000000010000000100000001000000010000000100000001000
00. Long: 232

         Cadena bits reducida:
         .0101000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000. Lon
g: 145

         Cadena bits con opcion:
         .1010100000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000. Lo
ng: 146

         Cadena de bit aumentada a esteganografiar:
         .1010100000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000. Lo
ng: 146
         INICIO DEL PROCESO DE ESTEGANOGRAFIADO
         (Esta operacion puede tardar unos segundos)

         FIN DEL PROCESO DE ESTEGANOGRAFIADO

         Fichero esteganografiado final: .ejemplos/taller2_Amp_Final.c.

                      (pulse intro para continuar)
```

Ilustración 112: Prueba de esteganografiado longitud cero para mayúsculas en código C ampliado

6. ESTEGOANÁLISIS DE LOS ALGORITMOS DE ESTEGANOGRAFIADO

En este punto llevaremos a cabo un estegoanálisis de los 3 modos de esteganografiado implementados y explicados anteriormente para analizar la invisibilidad y robustez de cada uno de ellos.

Como ya explicamos en el punto 2.5, *"El estegoanálisis es la disciplina dedicada al estudio de la detección de mensajes ocultos usando esteganografía"* y para ello, nos pondremos en la piel de un adversario que realizara un estegoanálisis con ataques activos y pasivos al estego-objeto para obtener o destruir su información oculta.

Para realizar los estegoanálisis, utilizaremos en todos los casos el mismo código base en C, dicho código corresponde a un programa en C que realiza todas las funciones necesarias para gestionar un taller.

Las características del código base son:

	Datos del fichero		
Nombre	Tamaño (bytes)	Nº Líneas	Nº Funciones
Taller2.c	32.768 bytes	1122 Líneas	24 Funciones

Tabla 29: Características del fichero

	Capacidad	Datos esteganografiables		Capacidad	
Método	Cap. Total	Caracteres	Mayúsculas	Capacidad	Capacidad %
Esteganografiado Código C	118 bits	14	23	0,105 bits/línea	10,517 %
Esteganografiado Ejecutable EXE	28 bits	3	5	0,025 bits/línea	2,496 %
Esteganografiado Código C Ampliado	146 bits	18	29	0,130 bits/línea	13,012 %

Tabla 30: Capacidades del fichero

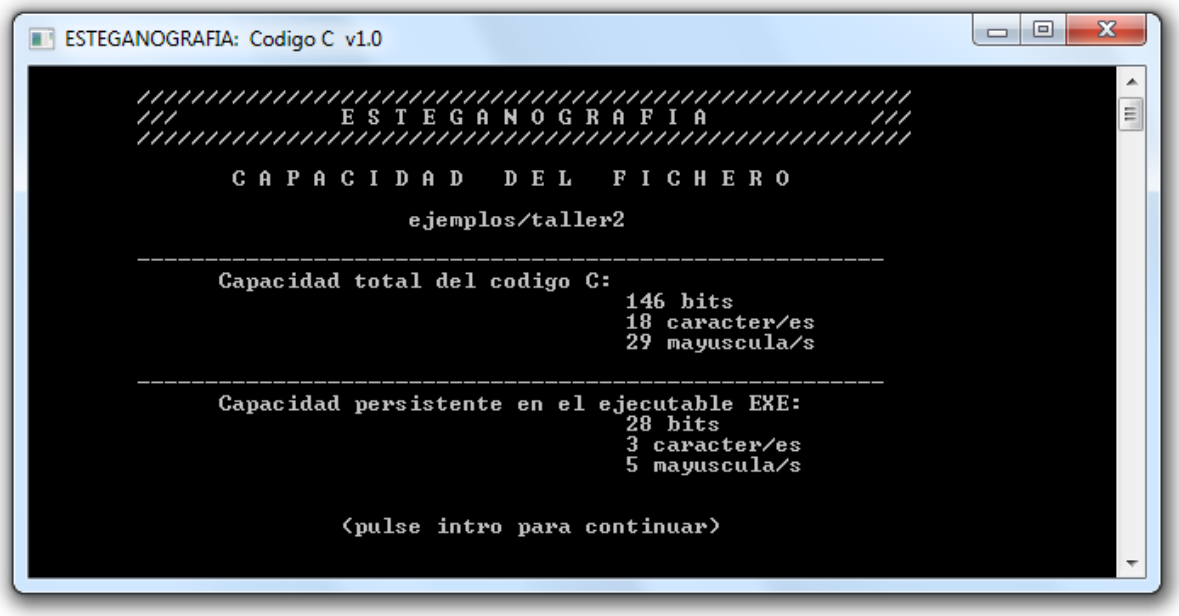


Ilustración 113: Resultado del análisis realizado por el programa de Esteganografiado.

6.1. ATAQUE ACTIVO

Este ataque se utiliza principalmente contra las técnicas de esteganografía de enmascaramiento y filtrado[17] utilizado con lo que es muy útil para este caso puesto que nuestro modo de esteganografiado se comporta como una marca de agua digital.

El objetivo de este ataque es inutilizar o eliminar la información adicional anexionada al código como puede ser la titularidad del autor o la información del usuario que ha adquirido los derechos de uso para así poder hacer un uso fraudulento del mismo.

Ataque activo al código C esteganografiado

Para este ataque utilizaremos como estego-objeto el código C del programa de gestión de talleres, comentado anteriormente, esteganografiado mediante el algoritmo de esteganografiado en ficheros de código .C en su modo de cadena de caracteres.

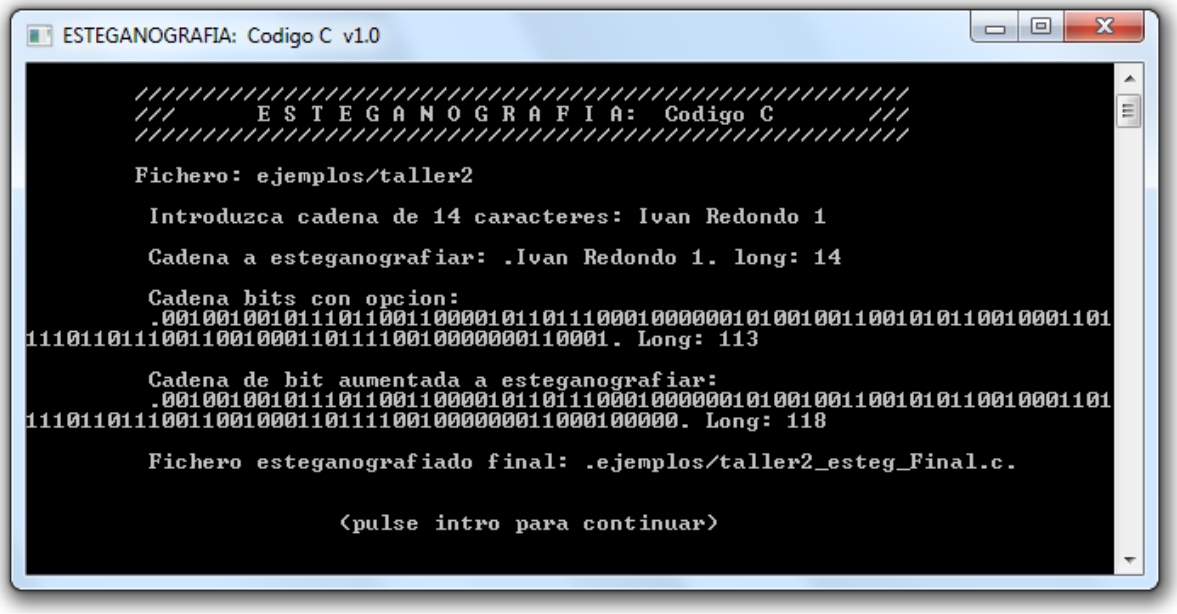


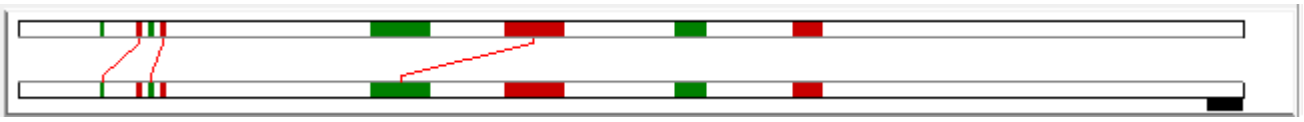
Ilustración 114: Esteganografiado mediante el algoritmo de esteganografiado en ficheros de código .c

Una vez obtenido el estego-objeto, lo modificamos para eliminar su información oculta.

Esta modificación del código ha de hacerse con cuidado, porque aunque la finalidad sea destruir la información esteganografiada, el código en c que forma el estego-objeto ha de seguir siendo funcional.

En este diagrama se pueden apreciar los puntos del fichero que se han modificado, en verde las partes poco modificadas y en rojo las partes muy modificadas, a mayor anchura de sección, mayor número de líneas modificadas y a su vez, las líneas rojas muestran partes del código cambiadas de sitio en el fichero.

FICHERO MODIFICADO



FICHERO ORIGINAL

[17] Explicado en el punto 2.2. TÉCNICAS DIGITALES DE ESTEGANOGRAFIADO

Una vez modificado el código, observamos que su desesteganografiado es incorrecto puesto que al haberse alterado algunas partes del código, el mensaje desesteganografiado es erróneo.

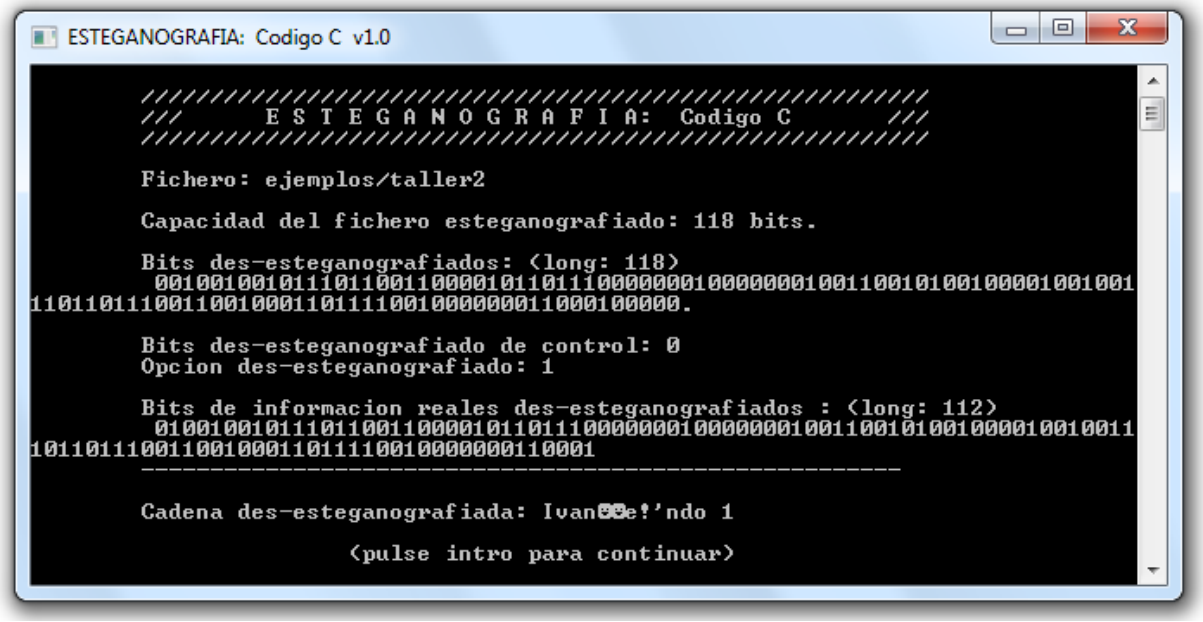


Ilustración 115: Desesteganografiado del estego-objeto en código C modificado

Cadena caracteres esteganografiada original:

- Ivan Redondo 1

Cadena bits esteganografiada original:

- 00100100101110110011000010110111100010000001010010011001010110010001101111011011100110010001101111001000000011000100000

Cadena caracteres desesteganografiada:

- Ivan ☻ ☻e!'ndo 1

Cadena bits desesteganografiada:

- 00100100101110110011000010110111100000001000000010011001010010000100100111011011100110010001101111001000000011000100000

Aun habiendo realizado varios cambios en el estego-objeto, muy pocos de ellos han tenido repercusión en la información oculta, tan solo los 9 bits marcados anteriormente en rojo. Esto hace que el mensaje desesteganografiado resultante sea distinto al original, pero aun reconocible. Esto es debido a que el mensaje original estaba formado por caracteres que a su vez están formados por un conjunto de 8bits, con lo cual cualquier alteración en uno de esos bits hace que se altere dicho carácter y se pierda la información esteganografiada.

Aunque el modo utilizado haya sido el de cadena de caracteres, el cual brinda al estego-objeto de mayor resistencia al estegoanálisis, se puede apreciar como un cambio en 9 de los 118 bits que forman el mensaje hace que este se vuelva ligeramente irreconocible obteniendo así un resultado positivo en este ataque.

Ataque activo al código C esteganografiado con el modo ampliado

Para este otro ataque, utilizaremos como estego-objeto el código C del programa de gestión de talleres, pero esta vez esteganografiado mediante el algoritmo de esteganografiado ampliado en ficheros de código .C y con el modo de cadena de mayúsculas.

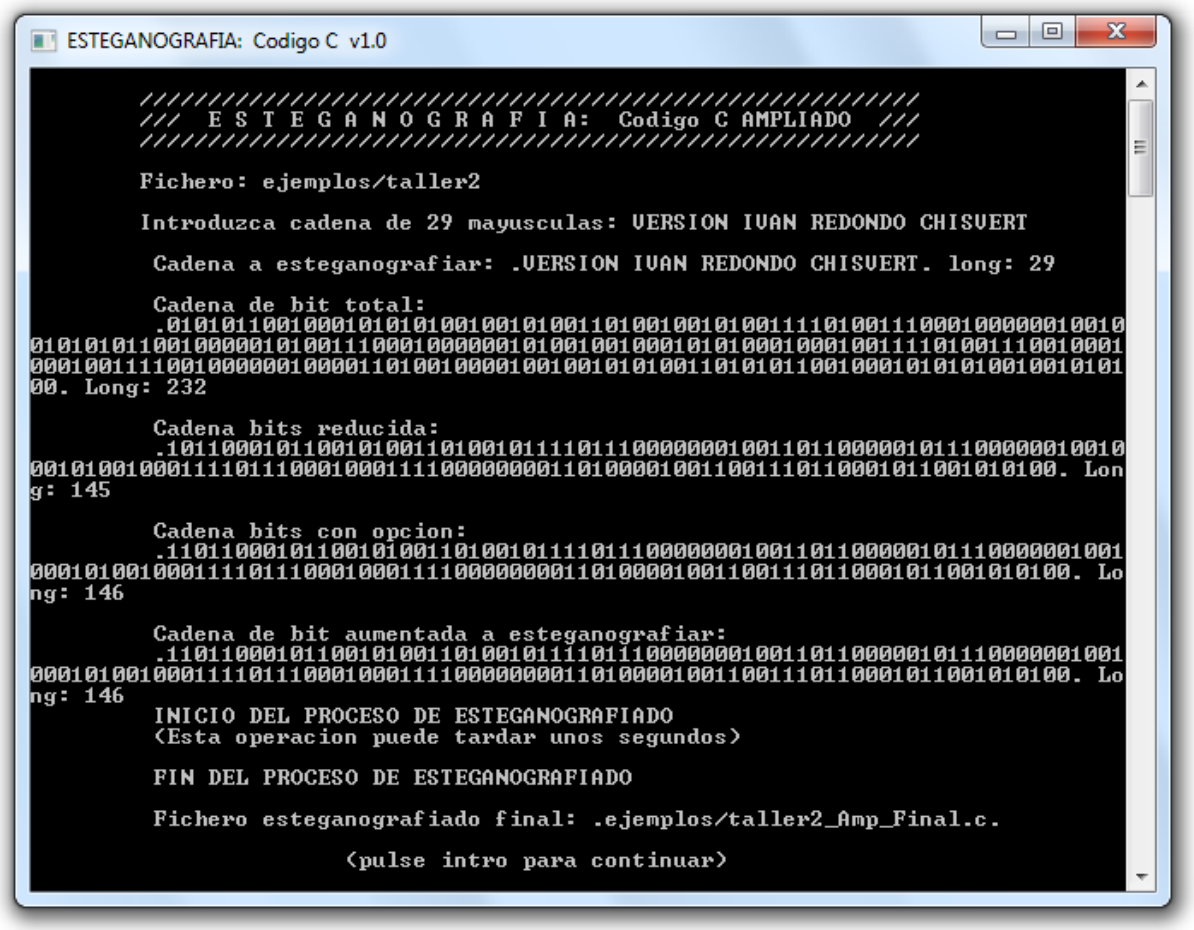
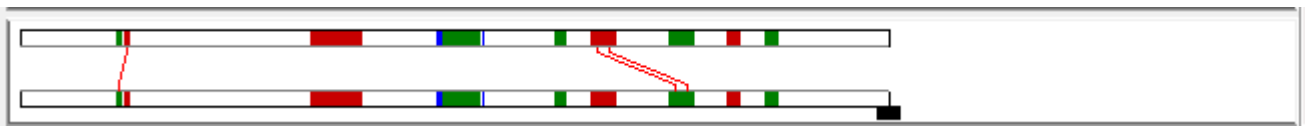


Ilustración 116: Esteganografiado mediante el algoritmo de esteganografiado ampliado

Una vez obtenido el estego-objeto, lo volvemos a modificamos para eliminar su información oculta, con cuidado de que el código en c que forma el estego-objeto siga siendo funcional.

En este diagrama se pueden apreciar los puntos del fichero que se han modificado, en verde las partes poco modificadas y en rojo las partes muy modificadas, a mayor anchura de sección, mayor número de líneas modificadas y a su vez, las líneas rojas muestran partes del código cambiadas de sitio en el fichero.

FICHERO MODIFICADO



FICHERO ORIGINAL

Una vez modificado el código, observamos otra vez que su desesteganografiado es incorrecto puesto que al haberse alterado algunas partes del código, el mensaje desesteganografiado es erróneo.

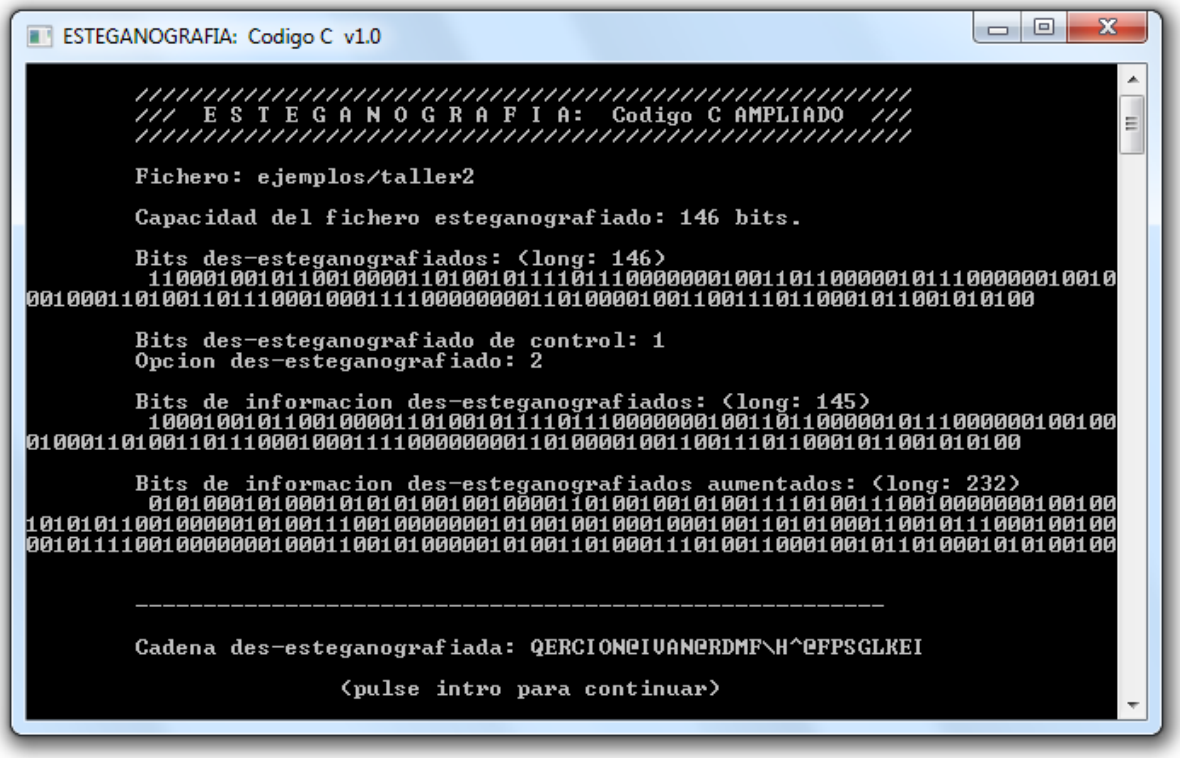


Ilustración 117: Desesteganografiado del estego-objeto en código C Ampliado modificado

Cadena caracteres esteganografiada:

- VERSION IVAN REDONDO CHISVERT

Cadena bits esteganografiada:

- 110110001011001010011010010111101110000000100110110000001011100000010010001010010001111011100010001111000000001101000010011001110110001011001010100

Cadena caracteres desesteganografiada:

- QERCION@IVAN@RDMF\H^@FPSGLKEI

Cadena bits desesteganografiada:

- 110001001011001010001101001011110111000000010011011000001011100000010010001000110100011011100010001111000000001101000010011001110110001011001010100

Esta vez, el modo de cadena de mayúsculas utilizado y a su vez el algoritmo de esteganografiado en código C ampliado, hacen que el estego-objeto resultante sea uno de los más sensibles al estegoanálisis. Esto es debido a que ahora, los caracteres del mensaje están formados por conjuntos de 3 bits con lo que cualquier cambio en el código es más posible que afecte a más caracteres. Por ello, se puede apreciar como un cambio menor que el anterior, esta vez de solo 9 de los 146 bits que forman el mensaje hace que este se vuelva prácticamente irreconocible obteniendo así otra vez un resultado positivo en este tipo de ataque.

Ataque activo al ejecutable EXE esteganografiado

En los apartados anteriores, hemos podido observar el comportamiento ante el estegoanálisis tanto de un estego-objeto resistente, el del esteganografiado en código C, como de otro muy sensible, el del esteganografiado en código C ampliado. Ambos métodos de esteganografiado son sensibles al estegoanálisis debido a que el estego-objeto, un fichero de código C, es fácilmente modificable por un atacante.

Esto ya no es posible en este ataque, donde utilizaremos como estego-objeto el ejecutable EXE del programa de gestión de talleres, esteganografiado mediante el algoritmo de esteganografiado en ejecutables EXE.

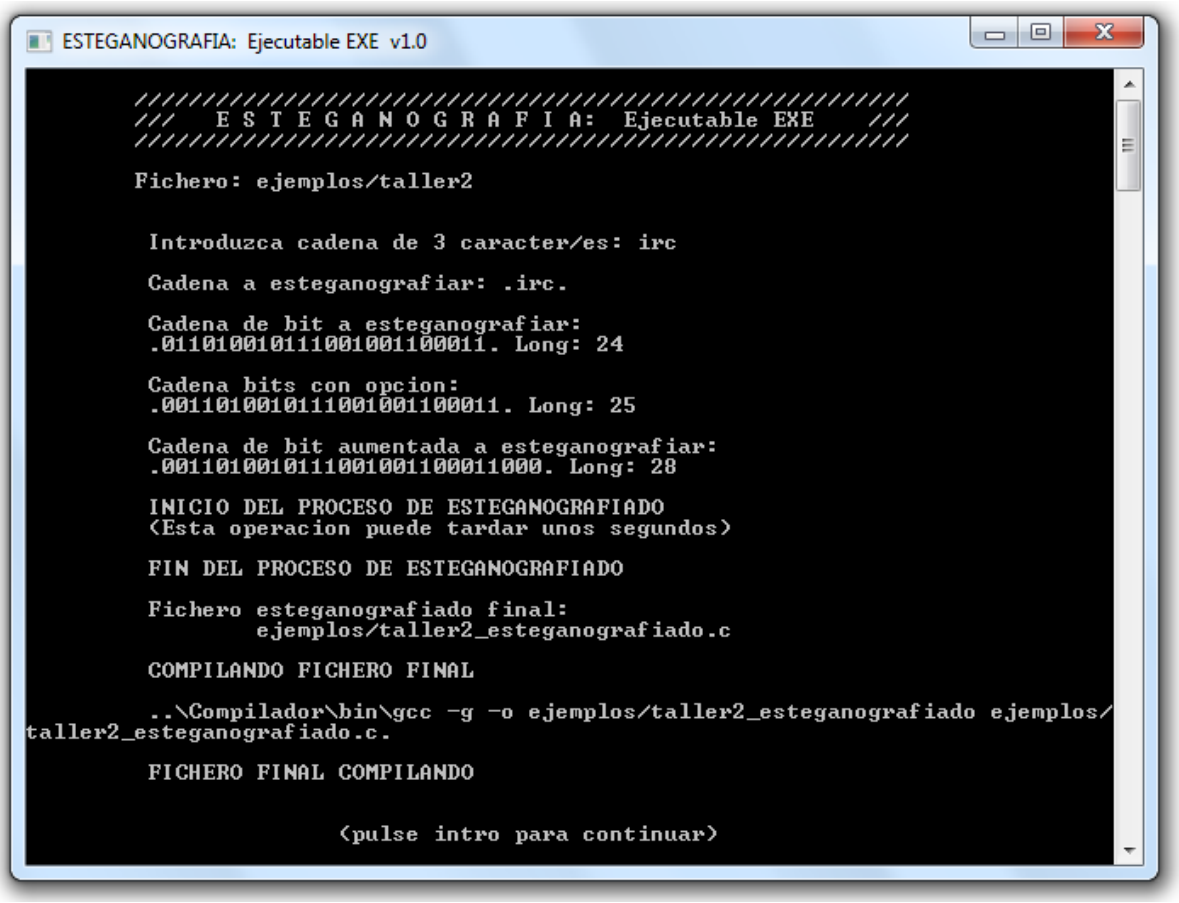


Ilustración 118: Esteganografiado mediante el algoritmo de esteganografiado en ejecutables .EXE

Cadena caracteres esteganografiada:

- irc

Cadena bits esteganografiada:

- 00110100101110010011000110000

Este método de esteganografía es totalmente resistente a un ataque activo debido a que es imposible modificar la información oculta en el ejecutable EXE sin que este pierda su funcionalidad.

6.2. ATAQUE PASIVO

En un ataque pasivo, la finalidad no es solo la detección del uso de la esteganografía, sino el intento de descifrado de la información esteganografiada en el estego-objeto original sin la modificación de este. Pero para considerar roto un sistema esteganográfico y considerarse un estegoanálisis pasivo como positivo solo se ha de detectar la existencia de un mensaje oculto y no el descifrado del propio mensaje.

Existen dos tipos principales de estegoanálisis pasivo:

Estegoanálisis pasivo manual:

Esta técnica se basa en una comparación de forma manual entre el portador y el estego-objeto en busca de diferencias que indiquen la existencia de datos ocultos. Para poder llevar a cabo esta comparación es necesario disponer de ambos ficheros, algo no muy común, y aun habiendo detectado cambios entre ambos ficheros, esto solo indicaría la existencia de datos ocultos, pero no recuperaría la información esteganografiada.

Como este método de esteganografiado[18] toma como base el código cero para esteganografiar en el la información, los únicos pares de esteganografiado que se verán modificados en el estego-objeto serán aquellos que cambien su valor de 0 a 1.

En esta huella de esteganografiado se puede ver una representación gráfica de los puntos de esteganografiado que posee el fichero portador.

Huella de esteganografiado del Código Cero (Fichero portador):

De todos los puntos de esteganografiado mostrados, al esteganografiar la información, solo se van a modificar algunos de ellos. En la siguiente imagen se pueden observar los puntos modificados del estego-objeto respecto a su código cero.

Pares de esteganografiado modificados en el Código resultante (Estego-objeto):

Conclusión, en este caso, al realizar el ataque pasivo manual comparando el fichero portador con uno de los estego-objeto generados, no es posible descubrir todos los puntos de esteganografiado puesto que solo se pueden observar en el código un pequeño número de cambios al parecer aleatorios. Con lo que se puede detectar que el estego-objeto contiene información oculta pero es imposible recuperarla si no se conoce el método de esteganografiado.

[18] Métodos explicados anteriormente en el apartado 3.ESTEGANOGRAFÍA EN CÓDIGO C

- **Ataque pasivo manual al código C esteganografiado**

En esta imagen podemos ver un ejemplo de la comparación entre el fichero portador y su estego-objeto generado por el método de esteganografiado en código C. En la imagen se puede apreciar una modificación en el código (marcado en verde).

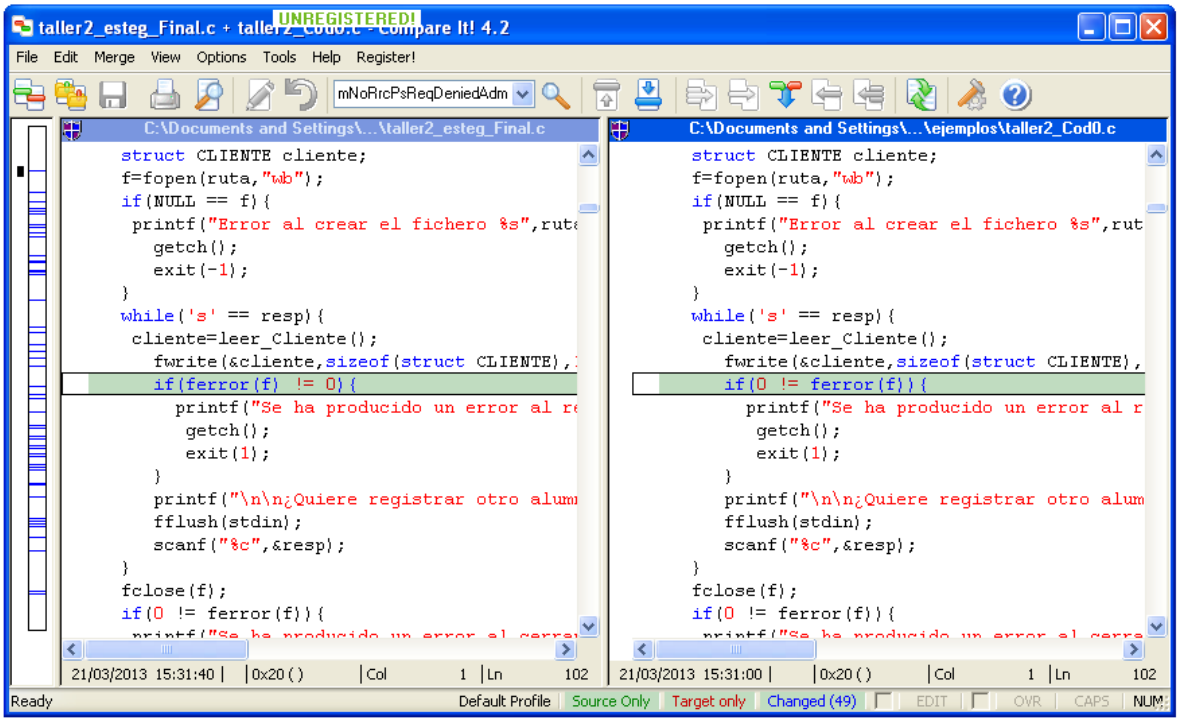


Ilustración 119: Comparación 1 entre el fichero portador y el estego-objeto esteganografiado en código C

Pero aun habiéndose detectado mediante el análisis dicha modificación en el código, donde presuntamente se esconde la información esteganografiada, a su vez, pasan totalmente inadvertidos al análisis otros tres puntos de esteganografiado.

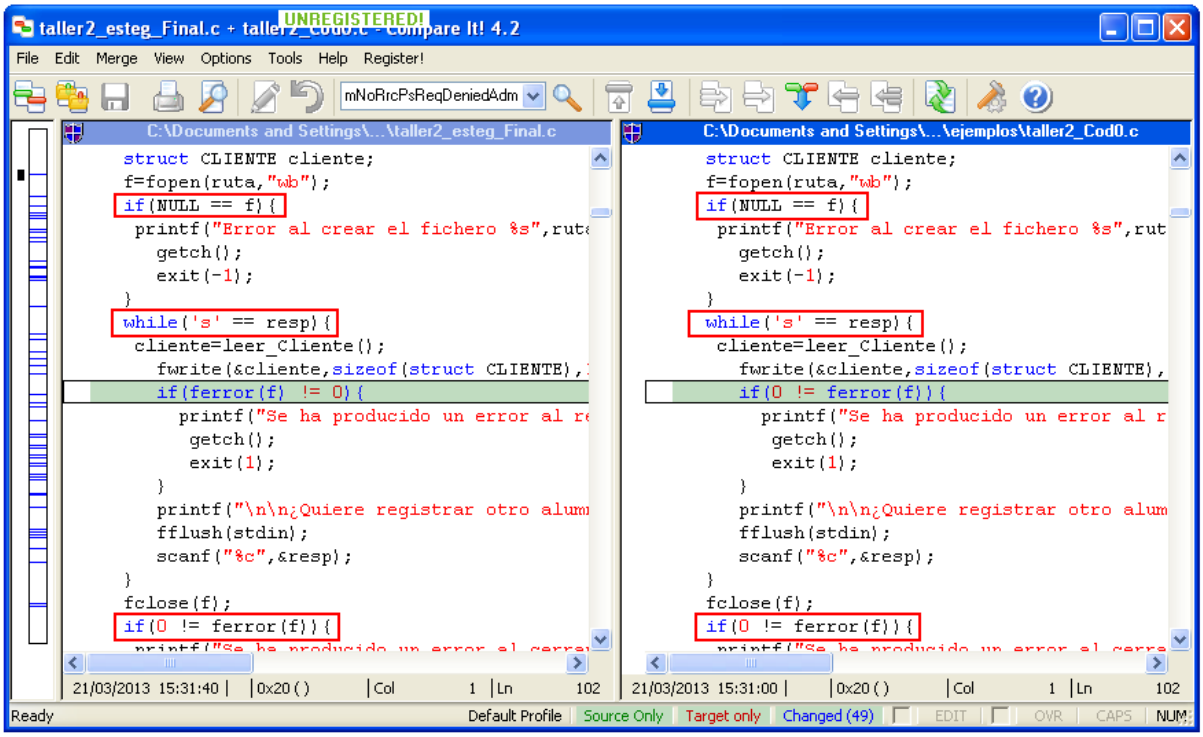


Ilustración 120: Comparación 2 entre el fichero portador y el estego-objeto esteganografiado en código C

- **Ataque pasivo manual al código C esteganografiado ampliado**

En esta otra imagen podemos ver ahora un ejemplo de la comparación entre el fichero portador y su estego-objeto generado por el método de esteganografiado en código C ampliado. En la imagen se pueden apreciar cuatro modificaciones en el código (marcado en verde).

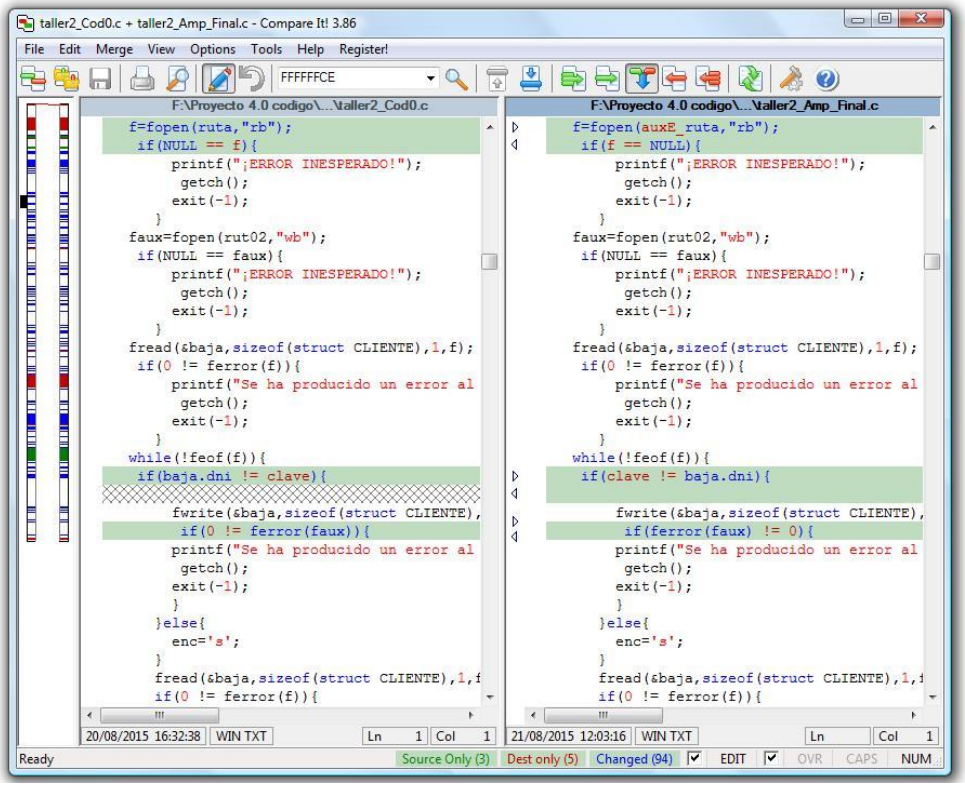


Ilustración 121: Comparación 1: Fichero portador y estego-objeto esteganografiado en código C ampliado

En este nuevo ataque, otra vez se aprecia como aun habiéndose detectado mediante el análisis modificaciones en el código, donde presuntamente se esconde la información esteganografiada, vuelven a pasar totalmente inadvertidos al análisis otros tres puntos de esteganografiado.

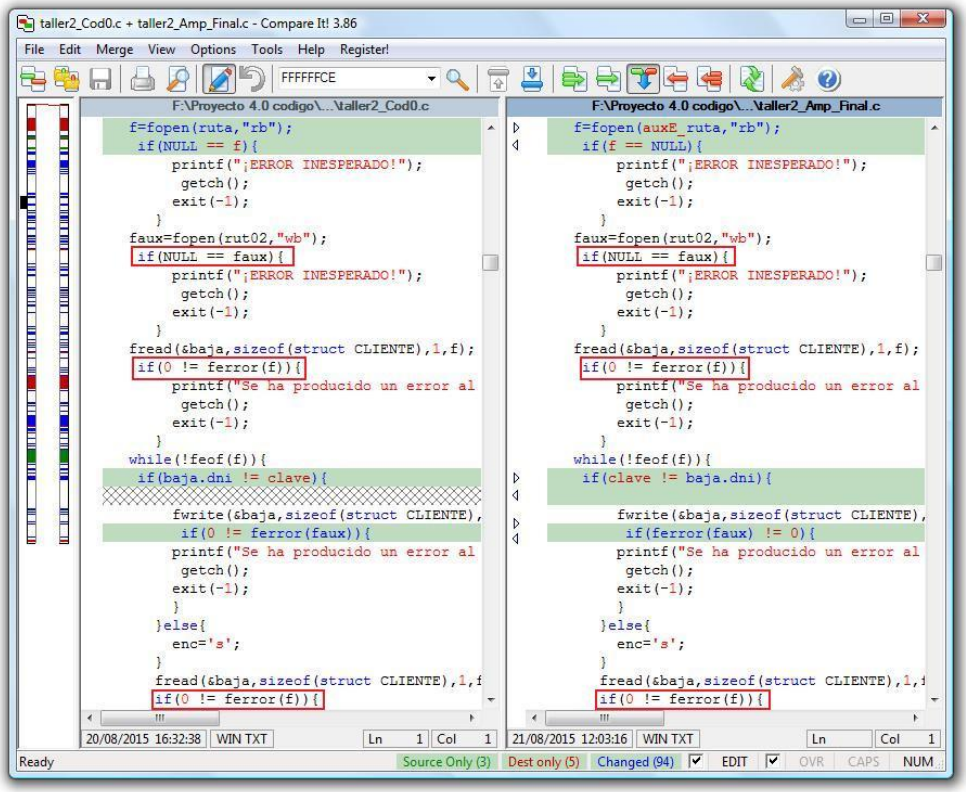


Ilustración 122: Comparación 2: Fichero portador y estego-objeto esteganografiado en código C ampliado

- **Ataque pasivo manual al ejecutable EXE esteganografiado**

En la siguiente imagen podemos ver ahora la comparación entre el ejecutable EXE portador y su estego-objeto generado por el método de esteganografiado en ejecutables EXE.

Todo lo comentado anteriormente para el ataque pasivo manual, ya no es aplicable para este último método de esteganografiado. Porque como se puede apreciar, la comparación entre los dos ejecutables nos devuelve que hay multitud de cambios donde si no se conoce el método de esteganografiado utilizado, no es posible apreciar si los cambios son producidos por información oculta o por el propio proceso de compilación.

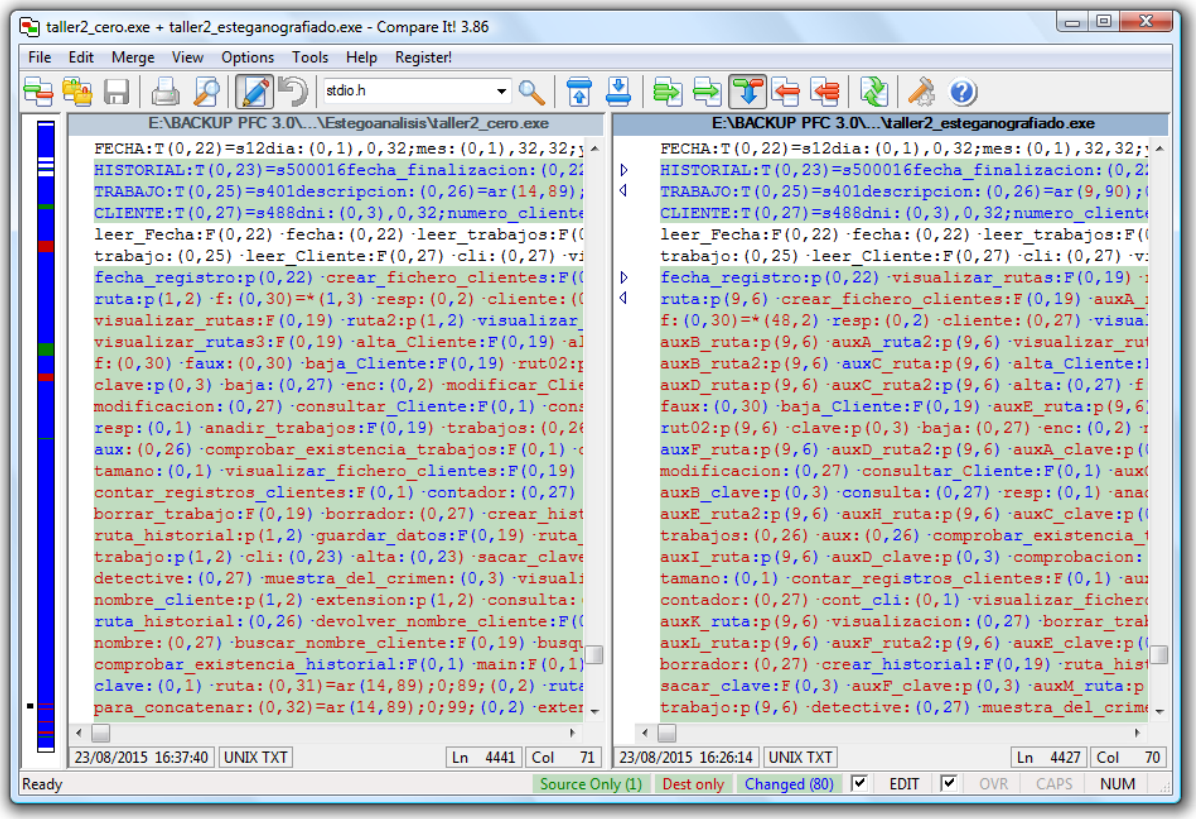


Ilustración 123: Comparación entre el ejecutable portador y el ejecutable estego-objeto esteganografiado

Estegoanálisis pasivo estadístico:

Esta técnica se basa en la utilización de software especializado que sin necesidad de tener el fichero portador, es capaz de buscar en el estego-objeto las huellas que dejan los programas más utilizados de esteganografiado para así detectar la existencia de información oculta.

En este caso en particular en el que se utiliza código C como estego-objeto, no existe un software especializado en su estegoanálisis, por lo que los mensajes son casi imposibles de encontrar para estos programas. Aunque el análisis de una gran cantidad de ficheros de código fuente podría dar información de cómo programa la gente y quizás así se podría detectar sino todas, si algunas de las huellas que deja este método de esteganografiado en el código y llegar a averiguar la existencia de "algo" en el código. Y debido a que para considerar roto un sistema esteganográfico solo se ha de detectar la existencia de un mensaje oculto, esto podría considerarse un estegoanálisis pasivo estadístico positivo.

Aunque esto no se encuentra dentro del ámbito del proyecto, vamos a realizar una simulación de cómo podría ser este estegoanálisis:

Vamos a partir del caso más sencillo y más evidente de detectar, el esteganografiado de expresiones en sentencias condicionales con operaciones relacionales: **IF (A == B)**

El resto de algoritmos y métodos de esteganografiado, si no imposibles son mucho más difíciles de detectar por este método de estegoanálisis.

Si tomamos que prácticamente toda la gente que programa lleva un orden en el momento de escribir las funciones el cual suele ser, primero la variable y luego el valor a comparar. Lo normal es encontrarnos sentencias condicionales del tipo "IF (variable == valor)" y no del tipo "IF (valor == variable)".

El siguiente código pertenece a un ejemplo esteganografiado con el programa y en el podemos observar 3 sentencias condicionales IF, una de ellas, la de la línea 7, la vemos como una sentencia normal y no nos llama la atención. Pero las otras dos, las de la línea 15 y 25, vemos que están escritas "al revés" lo cual a una persona que sepa programar le llama mucho la atención, y esto unido a que sospechamos que dicho código puede llevar algo oculto podemos considerar roto el sistema esteganográfico al haber detectado la existencia de un mensaje oculto.

```
1.  void crear_fichero_clientes(char *ruta)
2.  {
3.  FILE *f;
4.  char resp='s';
5.  struct CLIENTE cliente;
6.  f=fopen(ruta,"wb");
7.  if(f == NULL){
8.  printf("Error al crear el fichero %s",ruta);
9.  getch();
10. exit(-1);
11. }
12. while(resp == 's'){
13. cliente=leer_Cliente();
14. fwrite(&cliente,sizeof(struct CLIENTE),1,f);
15. if(0 != ferror(f)){
16. printf("Se ha producido un error al realizar la operacion");
17. getch();
18. exit(1);
19. }
20. printf("\n\n¿Quiere registrar otro alumno?: s/n");
21. fflush(stdin);
22. scanf("%c",&resp);
23. }
24. fclose(f);
25. if(0 != ferror(f)){
26. printf("Se ha producido un error al cerrar");
27. getch();
28. exit(1);
29. }
```

7. GESTIÓN DEL PROYECTO

En este apartado se aborda la gestión del proyecto atendiendo a su planificación y fases de ejecución y al presupuesto de los recursos empleados.

Para este apartado, se va a realizar una simulación debido a que por varios motivos los tiempos no se han podido ajustar a lo que aquí hay descrito. La duración real de este proyecto debido a un desarrollo intermitente por motivos personales y laborales ha sido de 6 años, realizándose la primera reunión con Don Jorge Blasco Alis el día 30 de Marzo de 2009.

7.1. PLANIFICACIÓN

La realización de este proyecto ha tenido una duración de 10 meses y puede dividirse en una serie de fases que se han ido ejecutando siguiendo un orden y atendiendo a una serie de dependencias entre ellas.

Las fases que componen la ejecución del proyecto son las siguientes:

IDENTIFICADOR	**Fase inicial.**
DEPENDENCIAS	-
DESCRIPCIÓN	Estudio de los distintos proyecto disponibles en el tablón de proyectos de la Universidad Carlos III de Madrid y elección de uno de ellos.
DURACIÓN (TOTAL COMPLETADO)	1 meses (10% PFC Realizado)

Tabla 31: Fase inicial

IDENTIFICADOR	**Fase de planificación y diseño.**
DEPENDENCIAS	Fase inicial.
DESCRIPCIÓN	Planificación del proyecto, estudio del estado del arte y diseño de los distintos algoritmos de esteganografiado.
DURACIÓN (TOTAL COMPLETADO)	2 meses (30% PFC Realizado)

Tabla 32: Fase de planificación y diseño

IDENTIFICADOR	**Fase de desarrollo y pruebas.**
DEPENDENCIAS	Fase de planificación y diseño.
DESCRIPCIÓN	Desarrollo, programación y prueba de cada uno de los algoritmos diseñados.
DURACIÓN (TOTAL COMPLETADO)	5 meses (80% PFC Realizado)

Tabla 33: Fase de desarrollo y pruebas

IDENTIFICADOR	**Fase de entrega.**
DEPENDENCIAS	Fase de desarrollo y pruebas.
DESCRIPCIÓN	Elaboración de la memoria del proyecto y de la presentación final para la defensa del proyecto.
DURACIÓN (TOTAL COMPLETADO)	2 meses (100% PFC Realizado)

Tabla 34: Fase de entrega

El siguiente diagrama de Gantt muestra las fases que componen la ejecución del proyecto:

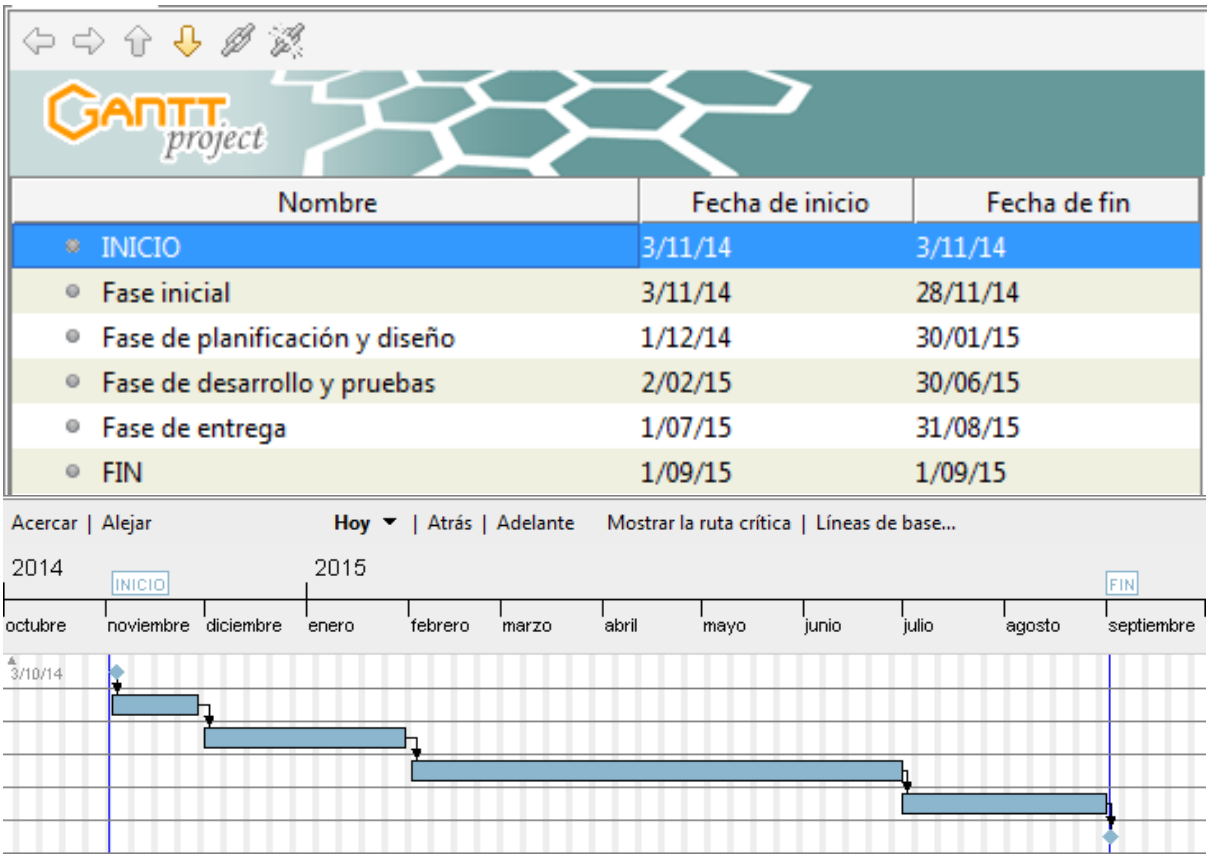


Ilustración 124: Diagrama Gantt

7.2. PRESUPUESTO

A continuación vamos a mostrar de manera detallada todas las partidas que forman el presupuesto del proyecto así como su coste.

RECURSOS PERSONALES

En las distintas fases de realización de este proyecto han intervenido las siguientes personas:

NOMBRE	CATEGORÍA	FUNCIÓN
Iván Redondo Chisvert	Ingeniero Técnico	Autor del proyecto
Jorge Blasco Alis	Ingeniero	Director del proyecto
Sergio Pastrana Portillo	Ingeniero	Tutor del proyecto

Tabla 35: Recursos personales del proyecto

El número de horas totales dedicadas al proyecto en cada parte y por cada uno de los recursos personales son las siguientes:

	Iván Redondo	Jorge Blasco	Sergio Pastrana
Fase inicial	40 horas	0 horas	0 horas
Fase de planificación y diseño	160 horas	50 horas	0 horas
Fase de desarrollo y pruebas	400 horas	0 horas	0 horas
Fase de entrega	160 horas	50 horas	30 horas
TOTAL	**760 horas**	**100 horas**	**30 horas**

Tabla 36: Horas por fase y recurso personal del proyecto

Por tanto el número de horas totales dedicadas al proyecto será la suma de las horas totales de cada uno de los recursos personales, siendo el coste total del proyecto de **890 horas**.

Para calcular el desglose presupuestario de los recursos personales empleados utilizaremos las tarifas para cada categoría establecidas en el *XVII Convenio colectivo nacional de empresas de ingeniería y oficinas de estudios técnicos*[19] y a los totales de horas empleadas por cada uno de los recursos personales del proyecto.

Nivel 1. Licenciados y titulados 2.º y 3.er ciclo universitario y Analista.
Salario anual: 23.618,28 € + Plus de Convenio 2.109,69 € = 25.727,97 €
Coste por hora: 25.727,97 € al año / 1.800 horas al año = 14,30 €

Nivel 2. Diplomados y titulados 1.er ciclo universitario. Jefe Superior.
Salario anual: 17.544,24 € + Plus de Convenio 2.109,69 € = 19.653,93 €
Coste por hora: 19.653,93 € al año / 1.800 horas al año = 10,92 €

El desglose presupuestario por cada uno recursos personales es el siguiente:

Recurso	Categoría	Dedicación (horas)	Coste (€/hora)	Coste Total (€)
Iván Redondo Chisvert	Ingeniero Técnico	760	10,92 €	8.300 €
Jorge Blasco Alis	Ingeniero	100	14,30 €	1.430 €
Sergio Pastrana Portillo	Ingeniero	30	14,30 €	429 €
			Total	**10.159 €**
			(TOTAL+21% IVA)	**12.292,39 €**

Tabla 37: Desglose presupuestario de los recursos personales

Por lo tanto, el importe total de los recursos personales empleados en el proyecto asciende a la cantidad de **doce mil doscientos noventa y dos con treinta y nueve euros (12.292,39 €)** impuestos incluidos.

[19] http://www.boe.es/boe/dias/2013/10/25/pdfs/BOE-A-2013-11199.pdf

RECURSOS TÉCNICOS

Como concepto de recursos técnicos, se muestran en la siguiente tabla, los equipos informáticos adquiridos así como su coste de amortización durante el proyecto. Todos los costes son calculados con los impuestos incluidos.

Calculo de la amortización:

$$\frac{A}{B} xCxD$$

A = Nº de meses desde la fecha de facturación (10 meses)
B = Periodo de depreciación (60 meses)
C = Coste del equipo
D = % de uso dedicado al proyecto

Descripción	Coste (€)	% Uso Dedicado al Proyecto	Coste Imputable (€)
Portátil acer Aspire 6930G	600€	50%	50€
Pendrive SanDisk 32Gb	12€	100%	2€
		TOTAL	**52€**

Tabla 38: Desglose presupuestario de los recursos técnicos

RECURSOS SOFTWARE

Por último, es necesario incluir en los gastos del proyecto el resto de costes directos derivados de su realización. En este caso, se incluyen, el precio del software que se ha utilizado y el precio de la conexión a internet que ha sido necesaria para recopilar información acerca del estado del arte, lenguajes y herramienta relacionados con el proyecto y también como medio de comunicación con el tutor. Todos los costes son calculados con los impuestos incluidos.

Recursos software: Sistema operativo Windows Vista y herramientas de ofimática de Microsoft, coste cero, software incluido en el ordenador en el momento de su compra. Programa de desarrollo, Dev-C++[20] y programas auxiliares, Notepad ++[21] y 7zip[22], todos software Open Source con coste cero para el proyecto.

Conexión a internet: Su uso, no se ha dedicado exclusivamente al proyecto, por lo que se ha estimado en un 30% el porcentaje de dedicación para la imputación.

Descripción	Coste (€)	% Uso Dedicado al Proyecto	Coste Imputable (€)
Software	0€	50%	0€
Conexión a internet	10 meses X 30€/mes = 300€	30%	90€
		TOTAL	**90€**

Tabla 39: Desglose presupuestario de los recursos software

[20] http://www.bloodshed.net/dev/
[21] https://notepad-plus-plus.org/
[22] http://www.7-zip.org/

PRESUPUESTO TOTAL

El presupuesto del proyecto de Esteganografía en código C tras la suma de todas las partidas, asciende a la cantidad de, **doce mil cuatrocientos treinta y cuatro con treinta y nueve euros (12.434,39 €)**, impuestos incluidos.

PARTIDA	COSTE
Recursos personales	12.292,39 €
Recursos técnicos	52,00 €
Recursos software	90,00 €
TOTAL	12.434,39 €

Tabla 40: Presupuesto del proyecto

8. CONCLUSIONES Y TRABAJOS FUTUROS

8.1. CONCLUSIONES

En este Proyecto de Fin de Carrera cabe destacar el estudio de las distintas técnicas actuales de esteganografiado existentes para distintos medios digitales, y de cómo utilizan éstas los estego-objetos. Tras este estudio, se ha investigado un nuevo medio para el estego-objeto, el código fuente escrito en lenguaje de programación "C". Se han podido descubrir una serie de puntos de esteganografiado donde ocultar finalmente la información.

A su vez, el estudio de la capacidad de esteganografiado y de su persistencia nos permitió diseñar no solo varios modos de esteganografiado como son el de caracteres y mayúsculas, sino que nos permitió diseñar también distintos algoritmos de esteganografiado orientados a solventar los distintos problemas que se fueron encontrando. Por ejemplo, el algoritmo de esteganografiado en ejecutables EXE acaba con el problema de la persistencia de la información esteganografiada tras la compilación del código escrito en lenguaje de programación "C" y el algoritmo de esteganografiado ampliado en ficheros de código "C" es capaz de aumentar la capacidad de esteganografiado inicial del código que se quiere esteganografiar.

De este modo, se puede concluir que el programa desarrollado cumple con todos los propósitos que se esperaban de él:

- **ESTEGANOGRAFIADO DE INFORMACIÓN:**
  Se obtiene la capacidad de ocultar información dentro del propio código.

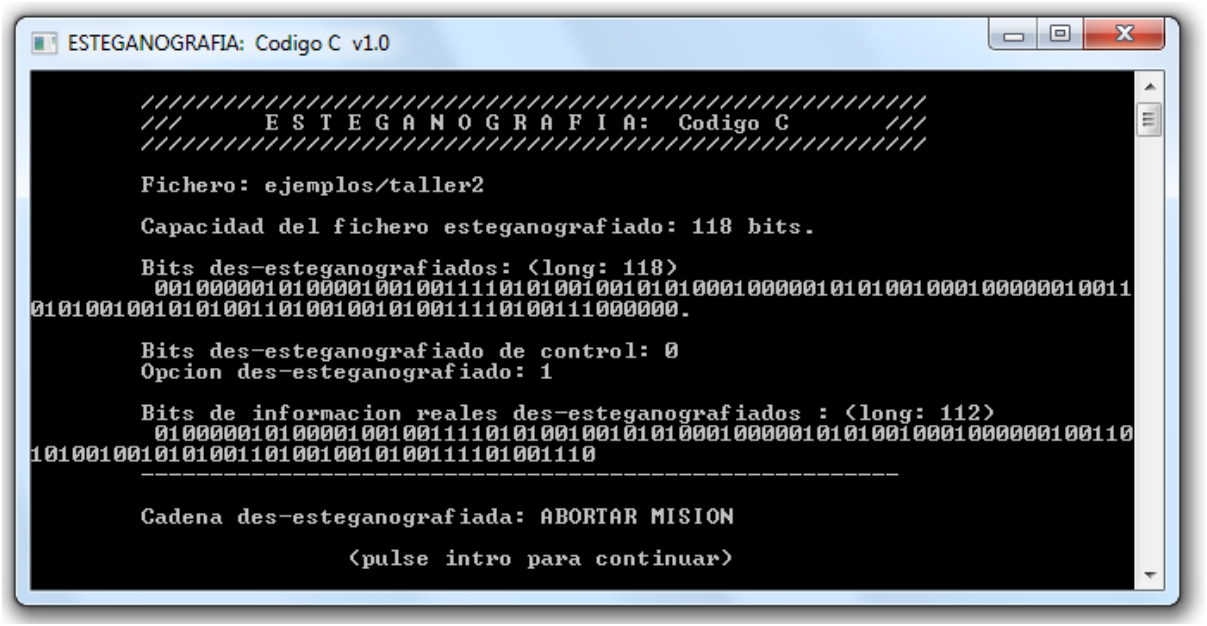


Ilustración 125: Ejemplo de esteganografiado de información secreta

- **FIRMA DIGITAL DE UN CÓDIGO:**

  Se obtiene la capacidad de firmar el código de manera oculta al usuario final del código.

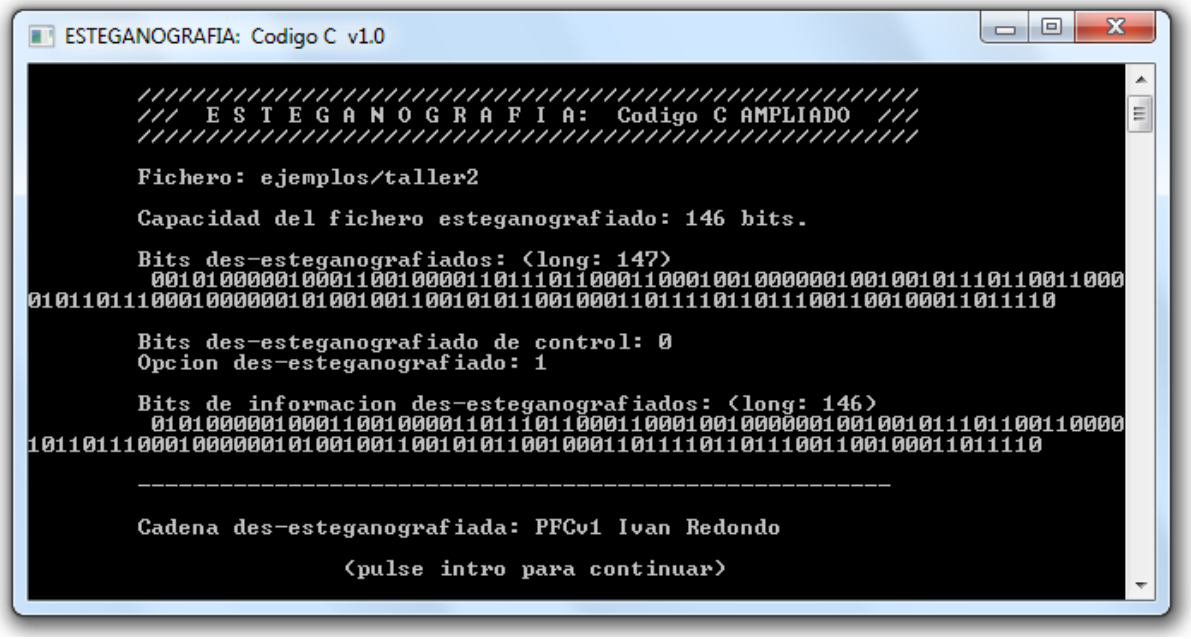


Ilustración 126: Ejemplo de firma digital de un código

- **IDENTIFICACIÓN DE DISTINTAS COPIAS DE UN MISMO CÓDIGO:**

  Se obtiene la capacidad de asignar cada una de las copias de un mismo código a distintos usuarios finales con el fin de poder identificarlas unívocamente.

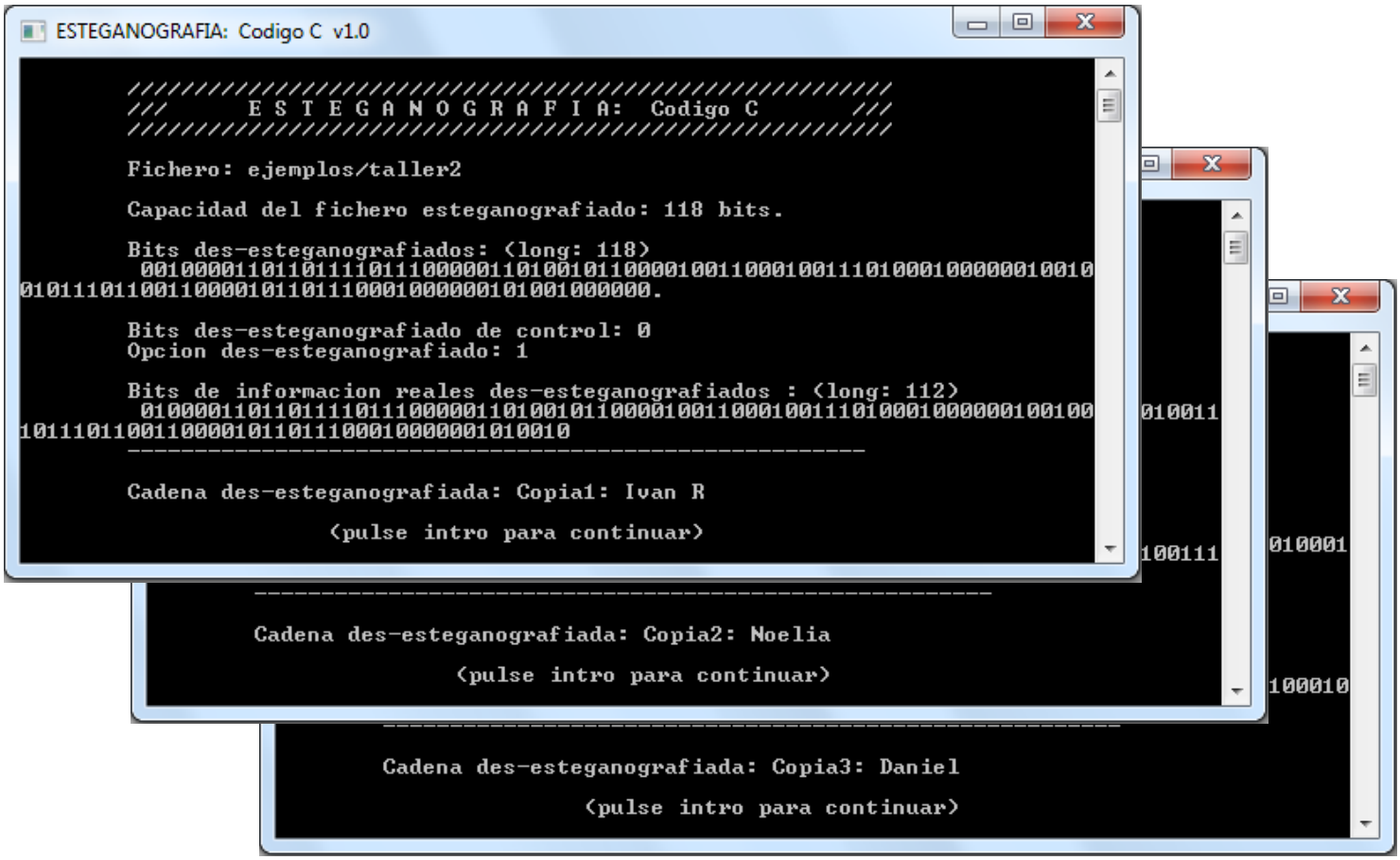


Ilustración 127: Ejemplo de identificación de distintas copias de un mismo código

- **IDENTIFICACIÓN DE UN CÓDIGO DENTRO DE OTRO:**
  Se obtiene la capacidad de firmar el código de manera oculta y de poder ser este identificado cuando forma a su vez, parte de un código mayor.

El programa original se llama taller2.C tiene una capacidad de 118 bits y está firmado con "Ivan Redondo Chisvert"

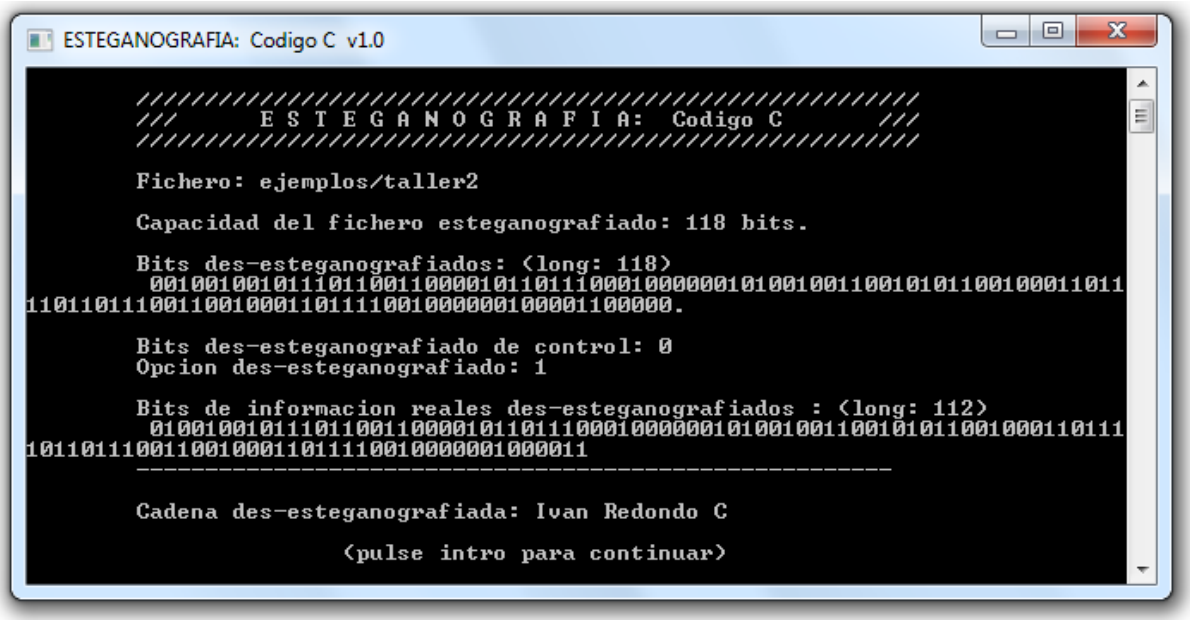


Ilustración 128: Ejemplo de código original firmado

La siguiente ilustración muestra un programa llamado Gestión.C con una capacidad de 164 bits donde al desesteganografiar se ha descubierto que parte de ese código pertenece a nuestro código Taller2.C y que está firmado con "Ivan Redondo Chisvert"

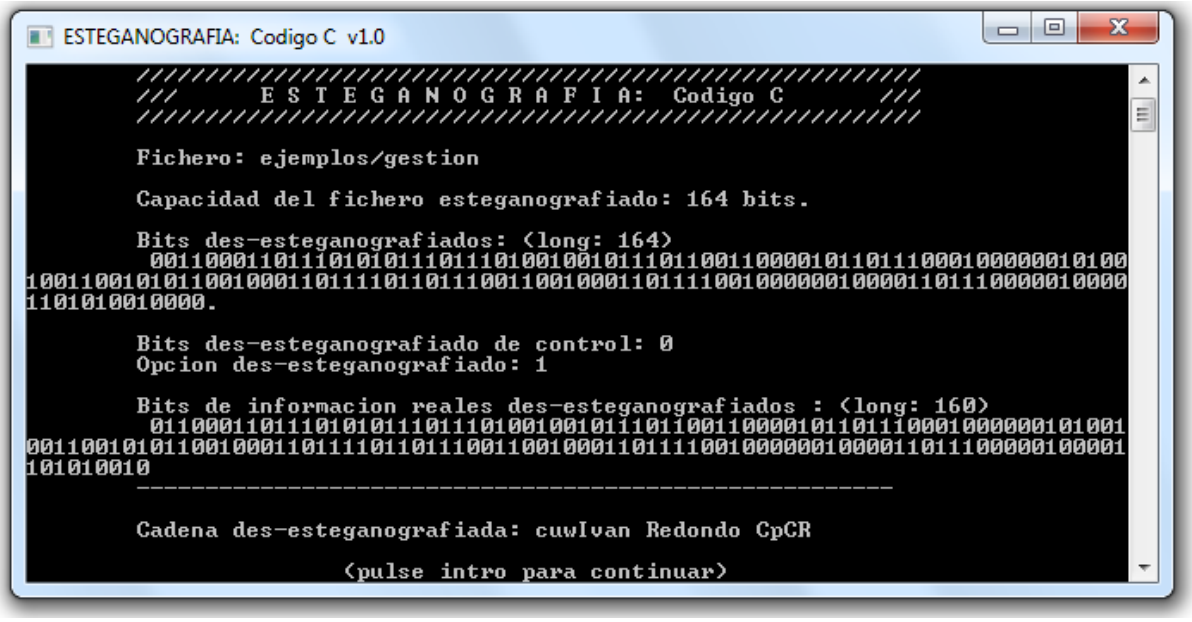


Ilustración 129: Ejemplo de código firmado detectado dentro de otro

Como conclusiones generales del proyecto destacar la dificultad encontrada a la hora de codificar la idea, lo que en concepto solo es cambiar de lado los operandos y los operadores de una formula se convierte en todo una odisea, cada código es un mundo y cada persona lo escribe de una manera distinta, dentro de los límites que tiene el compilador, sí, pero de mil maneras. Por lo tanto a un programa que es capaz de leer y reescribir un código y a su vez compilarle y descompilarle según lo necesite, se le ha tenido que, digámoslo de una manera coloquial, "enseñar" a programar.

Una vez resuelto este inconveniente, esto se convierte en su mayor ventaja y el punto principal de este proyecto, puesto que C es un lenguaje muy flexible y que permite programar con múltiples estilos esto es lo que permite su esteganografiado, tanto a la hora de incorporar información como a la de ocultarla.

Por último, para terminar con las conclusiones del proyecto, solo quedaría destacar su principal inconveniente, el progresivo desuso del código C ante Java. Pocos usos le quedan al código C fuera del ámbito educativo, por lo tanto la explotación en el ámbito real de lo expuesto y realizado en este proyecto no resulta muy atractiva

8.2. VALORACIÓN PERSONAL

Dejando a un lado las conclusiones del proyecto, comentar que en sus inicios, el proyecto que nos ocupa comenzó con la idea de investigar si sería posible la esteganografía de un código C, algo tan normalmente usado en la vida de un estudiante de informática como alejado de los medios o canales utilizados normalmente. Lo que se inició con un "mira a ver si esto se podría hacer", no tardó mucho en convertirse en un, "si es verdad que se puede hacer, ¡hagámoslo!" y así es como un puñado de "letras" y bucles en un folio se convirtieron en este proyecto.

Quien podría predecir que lo que se ve en esa imagen se acabaría convirtiendo en un proyecto de 8.750 líneas de código en C, en una memoria de más de 120 hojas y convirtiéndome a mí en una persona que ahora es incapaz de mirar un código sin ver cosas esteganografiadas en el, aunque no las tenga.

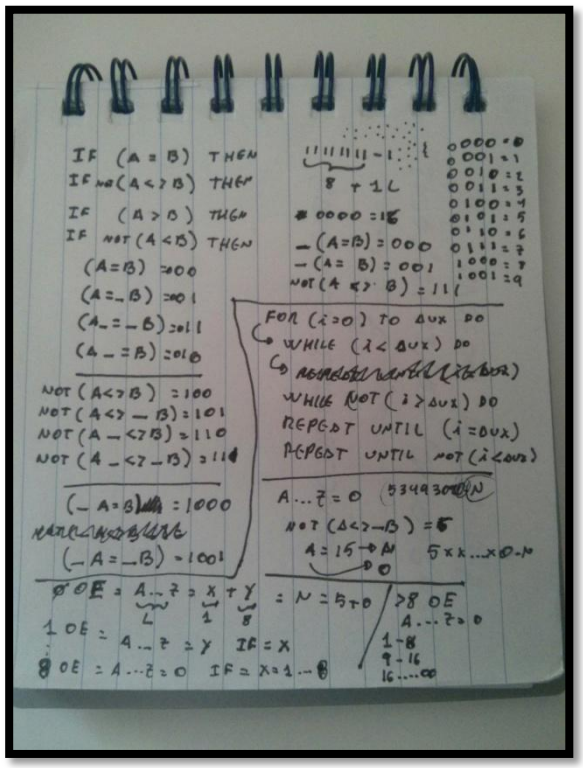


Ilustración 130: Inicio del proyecto

8.3. TRABAJOS FUTUROS

Al igual que a menudo surgen con posterioridad nuevas herramientas o campos de aplicación, que podrían haberse tenido en cuenta durante la realización de un proyecto y teniendo en cuenta que el objetivo principal de nuestro proyecto es investigar la posibilidad de esteganografiar un código C y que nuestra aplicación no es más que un prototipo que demuestra que esto si se pueden llevar a cabo. Esto deja la puerta abierta a multitud posibles trabajos futuros, pero solo destacaría dos grandes caminos a seguir.

- Lo comentado en el apartado anterior sobre el desuso progresivo del código C en el ámbito real, tratando de extrapolar lo aquí mostrado a distintos lenguajes de programación para así aumentar las funcionalidades.

- La mejora en el entorno visual del proyecto, incluyéndolo en una aplicación web o tal vez integrándolo en el propio compilador para conseguir así una mayor y más extendida utilización.

9. BIBLIOGRAFÍA

LIBROS:

- **Esteganografía y Estegoanálisis.**

Autores: Jordi Serra y Daniel Lerch.
Colaborador: Alfonso Muñoz
ISBN: 978-84-617-0021-9

- **Criptografía y Seguridad en Computadores.**

Manuel José Lucena López
Tercera Edición (Versión 2.10). Mayo de 2003.
http://www.uned.es/413042/material/Criptografia.pdf

- **"Esteganografía, el arte de ocultar información"**

Cuaderno de notas del Observatorio de la Seguridad de la Información.
Instituto Nacional de Tecnologías de la Comunicación
www.incibe.es

- **Los Nueve Libros de la Historia.**

Herodoto de Halicarnaso (484 A.C. - 425 A.C.).
Traducción. P. Bartolomé Pou, S. J. (1727-1802)
http://www.elaleph.com/

- **Leon Battista Alberti's Hypnerotomachia Poliphili**

Autor: Liane Lefaivre.
ISBN: 978-02-621-2204-7
https://mitpress.mit.edu/books/leon-battista-albertis-hypnerotomachia-poliphili

ARTÍCULOS Y APUNTES ONLINE:

- **Aula virtual de criptografía y seguridad de la información Crypt4you.**

Curso de privacidad y protección de comunicaciones digitales de la Universidad Politécnica de Madrid, "Lección 7. Canales subliminales. Esteganografía" por el Dr. Alfonso Muñoz, 02 de enero de 2014.
http://www.criptored.upm.es/crypt4you/temas/privacidad-proteccion/leccion7/leccion7.html

- **La Criptografía y otras técnicas históricas: Cómo ocultar información a ojos ajenos.**

Por Luis Enrique Corredera, 10 de noviembre de 2010
http://www.elreservado.es/news/view/220-noticias-espias/653-como-ocultar-informacion-a-ojos-ajenos

- **Protecting Your Images on the Web: A Look at Embedded Digital Watermarking vs. Digital Fingerprinting.**

Por Laura Evenson y Kyle Gundersen, 16 de marzo de 2010, para American Photographic Artists.
http://searchapa.us/wordpress/wordpress/?p=1407

- **The ancient art of hidden writing.**

Por Mark Ward, 2 de julio de 2010, para el área tecnológica de las noticias de la BBC.
http://www.bbc.com/news/10480477

ENLACES:

- **Licensestream**

LicenseStream Content Tracker: marcador digital de contenido para ayudar a sus propietarios a controlar y rentabilizar sus activos valiosos.
http://www.licensestream.com/licensestream2/Portal/solutions/content_tracker.aspx

- **Spammimic**

Web donde por medio de un algoritmo de creación esteganográfico y partiendo de un mensaje del usuario, se crea un correo de *spam* donde está oculto dicho mensaje.
http://spammimic.com/

- **Enscribe**

Software gratuito que permite crear una marca de agua en archivos de audio digital a partir de una imagen. Estas imágenes sólo pueden verse mediante un software de visualización de frecuencia y son visibles incluso después de la compresión del audio.
http://www.coppercloudmusic.com/enscribe/

- **Snow**

El programa SNOW se utiliza para ocultar mensajes en texto ASCII añadiendo espacios en blanco al final de las líneas. Debido a que generalmente en los editores de texto los espacios y tabuladores no son visibles, el mensaje queda oculto a observadores casuales. Y si se utiliza el cifrado incorporado, el mensaje no puede leerse si se detecta.
http://www.darkside.com.au/snow/